
COLLECTED WORKS
of A.M. TURING

MATHEMATICAL
LOGIC

R.O. Gandy[†] &
C.E.M. Yates
editors

NORTH-HOLLAND

Collected Works of A.M. Turing

MATHEMATICAL LOGIC

Collected Works of A.M. Turing

Mechanical Intelligence

Edited by D.C. INCE

Pure Mathematics

Edited by J.L. BRITTON

Morphogenesis

Edited by P.T. SAUNDERS

Mathematical Logic

Edited by R.O. GANDY and C.E.M. YATES

Collected Works of A.M. Turing

MATHEMATICAL LOGIC

Edited by

the late R.O. GANDY

and

C.E.M. YATES

Including prefaces by Solomon Feferman, Andrew Hodges,
Jack Good and Martin Campbell-Kelly



2001

ELSEVIER

AMSTERDAM • LONDON • NEW YORK • OXFORD • PARIS • SHANNON • TOKYO

ELSEVIER SCIENCE B.V.
Sara Burgerhartstraat 25
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

ISBN: 0-444-50423-0

© 2001 Elsevier Science B.V. All rights reserved.

This work is protected under copyright by Elsevier Science, and the following terms and conditions apply to its use:

Photocopying

Single photocopies of single chapters may be made for personal use as allowed by national copyright laws. Permission of the Publisher and payment of a fee is required for all other photocopying, including multiple or systematic copying, copying for advertising or promotional purposes, resale, and all forms of document delivery. Special rates are available for educational institutions that wish to make photocopies for non-profit educational classroom use.

Permissions may be sought directly from Elsevier Science Global Rights Department, PO Box 800, Oxford OX5 1DX, UK; phone: (+44) 1865 843830, fax: (+44) 1865 853333, e-mail: permissions@elsevier.co.uk. You may also contact Global Rights directly through Elsevier's home page (<http://www.elsevier.nl>), by selecting 'Obtaining Permissions'.

In the USA, users may clear permissions and make payments through the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA; phone: (+1)(978) 7508400, fax: (+1)(978) 7504744, and in the UK through the Copyright Licensing Agency Rapid Clearance Service (CLARCS), 90 Tottenham Court Road, London W1P 0LP, UK; phone: (+44) 207 631 5555; fax: (+44) 207 631 5500. Other countries may have a local reprographic rights agency for payments.

Derivative Works

Tables of contents may be reproduced for internal circulation, but permission of Elsevier Science is required for external resale or distribution of such material.

Permission of the Publisher is required for all other derivative works, including compilations and translations.

Electronic Storage or Usage

Permission of the Publisher is required to store or use electronically any material contained in this work, including any chapter or part of a chapter.

Except as outlined above, no part of this work may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior written permission of the Publisher. Address permissions requests to: Elsevier Science Global Rights Department, at the mail, fax and e-mail addresses noted above.

Notice

No responsibility is assumed by the Publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made.

Library of Congress Cataloging-in-Publication Data

Turing, Alan Mathison, 1912-1954.

Mathematical logic / edited by R.O. Gandy and C.E.M. Yates ; including prefaces by

Solomon Feferman ... [et al.]. -- 1st ed.

p. cm. -- (Collected works of A.M. Turing)

Includes bibliographical references.

ISBN 0-444-50423-0 (hardcover)

I. Logic, Symbolic and mathematical. I. Gandy, R. O. (Robin O.) II. Yates, C. E. M.

III. Title.

QA9.2 .T87 2001

511.3—dc21

2001023846

British Library Cataloguing in Publication Data

Mathematical logic. - (Collected works of A.M. Turing)

1. Turing, Alan, 1912-1954 2. Logic, Symbolic and mathematical

I. Gandy, R. O. II. Yates, C. E. M.

511.3

ISBN 0444504230

© The paper used in this publication meets the requirements of ANSI/NISO Z39.48-1992 (Permanence of Paper).

Printed in The Netherlands



ACKNOWLEDGEMENTS

- (1) The following articles by Alan Turing which appear on pages [18]–[53], [54]–[56] and [81]–[148] of this book, have been reproduced by kind permission of the London Mathematical Society from: P. Lond. Math. Soc. series 2 vol. 42 (1937) pp. 230–265 (On computable numbers, with an application to the Entscheidungsproblem); ibid. vol. 43 (1937) pp. 544–546 (On computable numbers, with an application to the Entscheidungsproblem. A correction) and P. Lond. Math. Soc. Series 2 vol. 45 (1939) pp. 161–228 (Systems of logic based on ordinals).
- (2) As indicated on pages [59], [70], [158], [168] and [186], the following articles have been printed with permission of the Association for Symbolic Logic from: J. Symb. Log. vol. 2 (1937) pp. 153–163 (Computability and λ -definability); ibid. p. 164 (The p -function in λ - K conversion), J. Symb. Log. vol. 7 (1942) pp. 28–33 (A formal theorem in Church's theory of types). J. Symb. Log. vol. 7 (1942) pp. 146–156 (The use of dots as brackets in Church's system) and J. Symb. Log. vol. 13 (1948) pp. 80–94 (Practical forms of type theory).
- (3) The text of the memoir on Alan Mathison Turing by M.H.A. Newman, which appears on pages [269]–[279], has been reproduced from the Bibliographical Memoirs of Fellows of The Royal Society vol. 1 (November 1955) pp. 253–263 by kind permission of The Royal Society and of Edward and William Newman, the author's executors. The source material for reproducing the photograph accompanying the memoir (on page [268]) is by courtesy of the National Portrait Gallery (London).

Finally, acknowledgement is gratefully made to Ros Moad at the Turing Digital Archive for making available source material for the frontispiece photograph of Alan Turing taken during the thirties. The original is located at King's College Library, Cambridge (copyright holder unknown).

PREFACE

It is not in dispute that A.M. Turing was one of the leading figures in twentieth-century science. The fact would have been known to the general public sooner but for the Official Secrets Act, which prevented discussion of his wartime work. At all events it is now widely known that he was, to the extent that any single person can claim to have been so, the inventor of the “computer”. Indeed, with the aid of Andrew Hodges’s excellent biography, *A.M. Turing: the Enigma*, even non-mathematicians like myself have some idea of how his idea of a “universal machine” arose – as a sort of byproduct of a paper answering Hilbert’s *Entscheidungsproblem*. However, his work in pure mathematics and mathematical logic extended considerably further; and the work of his last years, on morphogenesis in plants, is, so one understands, also of the greatest originality and of permanent importance.

I was a friend of his and found him an extraordinarily attractive companion, and I was bitterly distressed, as all his friends were, by his tragic death – also angry at the judicial system which helped to lead to it. However, this is not the place for me to write about him personally.

I am, though, also his legal executor, and in fulfilment of my duty I have organised the present edition of his works, which is intended to include all his mature scientific writing, including a substantial quantity of unpublished material. The edition will comprise four volumes, i.e.: *Pure Mathematics*, edited by Professor J.L. Britton; *Mathematical Logic*, edited by Professor R.O. Gandy and Professor C.E.M. Yates; *Mechanical Intelligence*, edited by Professor D.C. Ince; and *Morphogenesis*, edited by Professor P.T. Saunders.

My warmest thanks are due to the editors of the volumes, to the modern archivist at King’s College, Cambridge, to Dr. Arjen Sevenster and Mr. Jan Kastelein at Elsevier (North-Holland), and to Dr. Einar H. Fredriksson, who did a great deal to make this edition possible.

P.N. FURBANK

ALAN MATHISON TURING – CHRONOLOGY

- 1912 Born 23 June in London, son of Julius Mathison Turing of the Indian Civil Service and Ethel Sara née Stoney
- 1926 Enters Sherborne School
- 1931 Enters King's College, Cambridge as mathematical scholar
- 1934 Graduates with distinction
- 1935 Is elected Fellow of King's College for dissertation on the Central Limit Theorem of Probability
- 1936 Goes to Princeton University where he works with Alonzo Church
- 1937 (January) His article "On Computable Numbers, with an Application to the Entscheidungsproblem" is published in *Proceedings of the London Mathematical Society*
Wins Procter Fellowship at Princeton
- 1938 Back in U.K. Attends course at the Government Code and Cypher School (G.C. & C.S.)
- 1939 Delivers undergraduate lecture-course in Cambridge and attends Wittgenstein's class on Foundations of Mathematics
4 September reports to G.C. & C.S. at Bletchley Park, in Buckinghamshire, where he heads work on German naval "Enigma" encoding machine
- 1942 Moves out of naval Enigma to become chief research consultant to G.C. & C.S.
In November sails to USA to establish liaison with American code-breakers
- 1943 January–March at Bell Laboratories in New York, working on speech-encypherment
- 1944 Seconded to the Special Communications Unit at Hanslope Park in north Buckinghamshire, where he works on his own speech-encypherment project *Delilah*
- 1945 With end of war is determined to design a prototype "universal machine" or "computer". In June is offered post with National Physical Laboratory at Teddington and begins work on ACE computer
- 1947 Severs relations with ACE project and returns to Cambridge
- 1948 Moves to Manchester University to work on prototype computer
- 1950 Publishes "Computing Machinery and Intelligence" in *Mind*
- 1951 Is elected FRS. Has become interested in problem of morphogenesis
- 1952 His article "The Chemical Basis of Morphogenesis" is published in *Philosophical Transactions of the Royal Society*
- 1954 Dies by his own hand in Wimslow (Cheshire) (7 June)

PREFACE TO THIS VOLUME

Although the unpublished papers were left to Robin Gandy in Turing's will, the matter of inquiring into the possibility of their publication seems to have rested in the first instance with Max Newman¹. Newman ran into problems with the assessment of some of the unpublished papers, and then on retiring in 1963 he left the whole matter in Robin's hands. It was in fact far too big a task for one person but that was not truly realised until 1988 when Professor Furbank, Turing's executor and overall editor of the Collected Works, stepped in to take a new initiative. There followed an intensive period of activity in which he brought in three additional editors. They did a remarkable job and their volumes were published in 1992.

So why has this, the fourth volume, taken so much longer? As often there is no single reason, but some explanation is due. It was planned at the same time as the other three, becoming the only part of the Collected Works to be edited by Robin. In 1991 I was invited to help him and we made progress, though as expected it was slow. I should at this point make two things clear. First, the scope of the volume envisaged at that point covered only what is now in Parts I and II. Secondly, Robin had in fact written very careful first drafts of prefaces for all of the papers except the 'Ordinal Logics' paper as far back as 1978! He wished to rewrite them and that is what took time. He particularly wanted to rewrite the preface to the famous 1937 paper and to tackle the 'Ordinal Logics' paper. Since Robin had no draft preface for the latter, I did at that time initiate correspondence with Professor Solomon Feferman with a view to possibly using extracts from his excellent paper 'Turing in the Land of $O(z)$ ' for this purpose.

Then very sadly, Robin died unexpectedly in November 1995, and so I had to take a fresh look at the volume. I have been especially grateful to Professor Feferman for his prefaces, and for overall advice. Turing's biographer, Dr. Andrew Hodges, has also been particularly helpful. He suggested that this was an opportunity to take some new directions and tidy up some loose ends – hence Part III – and he subsequently provided two prefaces for Part III, including that for the elusive *Enigma* paper written probably in 1940 but not released until 1996.

¹ Professor Maxwell Hermann Alexander Newman, FRS. Newman was involved at certain crucial times in Turing's life. He introduced him to logic in 1935, was also at Bletchley Park and finally brought him to Manchester in 1948. He wrote Turing's Biographical Memoir for the Royal Society, which can be found in this volume.

I am grateful to Professor Jack Good, who worked with Turing at Bletchley Park, and Dr. Martin Campbell-Kelly for the other two prefaces in Part III.

Finally, I must thank Professor Nick Furbank and Dr. Arjen Sevenster for their help and forbearance in what has been a very protracted task.

Mike YATES
July 1999

CONTENTS

Preface	vii
Alan Mathison Turing – Chronology	viii
Preface to this volume	ix
Part I Computability and Ordinal Logics	1
Historical Introduction (Solomon Feferman)	3
1937 On Computable Numbers, with an Application to the Entscheidungsproblem (<i>P. Lond. Math. Soc. (2)</i> 42 , 230–265)	
1937 – A correction (<i>ibid.</i> 43 , 544–546)	
Preface	9
Turing text	18
A correction	54
1937 Computability and λ -definability. (<i>J. Symb. Log.</i> 2 , 153–163)	
1937 The p -function in $\lambda - K$ Conversion (<i>J. Symb. Log.</i> 2 , 164)	
Preface	57
Turing texts	59
1938 Systems of Logic based on Ordinals (<i>P. Lond. Math. Soc. (2)</i> 45 , 161–228)	
Preface (Solomon Feferman)	71
Turing text	81
Part II Type Theory	149
General Introduction to Turing’s work on Type Theory	151
Published papers	
1942 (with M.H.A. Newman) A Formal Theorem in Church’s Theory of Types (<i>J. Symb. Log.</i> 7 , 28–33)	
Preface	155
Turing – Newman text	158
1942 The Use of Dots as Brackets in Church’s System (<i>J. Symb. Log.</i> 7 , 146–156)	
Preface	165
Turing text	168

1948	Practical Forms of Type Theory (<i>J. Symb. Log.</i> 13 , 80–94)		
	Preface	179	
	Turing text	186	
Unpublished papers			
1941	Some Theorems about Church’s System (three unpublished manuscripts)	201	
1943–4	Practical Forms of Type Theory II	207	
1944–5	The Reform of Mathematical Notation	211	
Part III – Enigmas, Mysteries and Loose Ends		223	
Turing’s Treatise on the Enigma			
	Preface (Andrew Hodges)	225	
	Excerpts from the “Enigma Paper”	230	
Turing’s Papers on Programming			
	Preface (Martin Campbell-Kelly)	243	
	Excerpts from original paper	251	
Minimum Cost Sequential Analysis			
	Excerpt from unpublished manuscript	255	
	Commentary (Jack Good)	255	
The Nature of Turing and the Physical World (Andrew Hodges)			259
Letter from Robin Gandy to Max Newman			265
Royal Society Memoir (Max Newman)			268
Bibliography			281
Lists of contents of other volumes			289
Appendix: matters arising from other volumes			293

Part I

Computability and Ordinal Logics

The historical introduction which opens Part I has been adapted from Feferman's excellent paper [1988]: 'Turing in the Land of $O(z)$ '. The editor wishes to reiterate his gratitude to both Professor Feferman and Oxford University Press, the original publisher of the volume Herken [1988] which includes that paper.

This Page Intentionally Left Blank

Historical Introduction

by Solomon Feferman

The story of how Turing came to write his papers on computable numbers and ordinal logics is contained in Andrew Hodges' excellent biography, *Alan Turing, The Enigma* [1983]. It is retold in the following in condensed form, drawing extensively on Hodges for the relevant biographical details, as well as Kleene [1981] and Feferman [1986] for the development of logic and recursion theory in this period.

Paper 1. On Computable Numbers with an Application to the Entscheidungsproblem

The story begins with Turing's major achievement, his work on computability, carried out in 1935–6 soon after he became a fellow of King's College Cambridge at the age of 23. In the Spring of 1935 Turing attended a course on the Foundations of Mathematics given by the topologist M.H.A. Newman. Among other things, Newman explained Hilbert's problems concerning consistency, completeness and decidability of various axiomatic systems, as well as Gödel's incompleteness results for sufficiently strong such systems. Turing had already been interested in mathematical logic but had been working primarily on other areas of mathematics, especially group theory. Newman's course served to focus his interests in logic; in particular, Turing became intrigued by the *Entscheidungsproblem* (decision problem) for the first order predicate (or functional) calculus, and this came to dominate his thought from the summer of 1935 on (Hodges [1983], p. 94). In grappling with this problem he was led to conclude that the solution must be negative; but in order to demonstrate that, he would have to give an exact mathematical analysis of the informal concept of *computability by a strictly mechanical process*. This Turing achieved by mid-April 1936, when he delivered a draft of **Paper 1** to Newman. At first Newman was skeptical of Turing's analysis, thinking that nothing so straightforward in its basic conception as the Turing machines could be used to answer this outstanding problem. However, he finally satisfied himself that Turing's notion did indeed provide the most general explanation of finite mechanical process, and he encouraged the paper's publication.

Neither Newman nor Turing were then aware that the question of analyzing the notion of *effective calculability* had occupied the attention of Gödel, Herbrand, and especially Church since the early 1930's. This side of the story is well-told in Kleene [1981], on the origins of recursive function theory.¹ Kleene

¹ Cf. also Gödel [1986], pp. 338–345 and Davis [1982].

was a Ph.D. student of Church from 1931 to 1933 (along with Rosser). Church was promoting a universal system for logic and mathematics in the framework of the lambda (λ)-symbolism for defining functions, and set Kleene the problem of developing the theory of positive integers in his formalism, using an identification of the integers with certain λ -terms. The initial steps were rather difficult (even the predecessor function posed a problem), but once the first hurdles were cleared, Kleene was able to show more and more number-theoretic functions definable by the conversion processes of λ -terms. But Church's original system was shown before long (by Kleene and Rosser in 1934) to be inconsistent, and attention was then narrowed to a demonstrably consistent subsystem, which came to be called the λ -calculus.² The consistency of this subsystem was established by Church and Rosser, with some input by Kleene. As it happened, Kleene had already achieved all of his previous definability results in this restricted calculus. It was clear from the Church–Rosser consistency property that every λ -definable function (in the sense given by convertibility in the λ -calculus) is effectively calculable; moreover, Kleene was able to show each example of an effectively calculable function, which came to mind, to be λ -definable. In view of Kleene's success in meeting all such challenges, Church was led to make the proposal, which has come to be known as *Church's Thesis*, that the λ -definable functions comprise all the effectively definable functions. In the words of Kleene: "When Church proposed the thesis, I sat down to disprove it by diagonalising out of the class of the λ -definable functions. But quickly realizing that the diagonalisation cannot be done effectively, I became overnight a supporter of the thesis." (Kleene [1981], p. 59).

Gödel (coming from the University of Vienna) visited the Institute for Advanced Study in Princeton during the year 1933–34, and in the spring of 1934 he gave lectures on his incompleteness results. Notes for these lectures were taken by Kleene and Rosser and, after corrections by Gödel, were circulated at the time. They were subsequently reproduced in Davis' collection [1965] of basic papers on the "undecidable" and computable functions, and more recently in Volume I of Gödel's *Collected Works*. During the course of these lectures, Gödel presented a definition, based on a suggestion of Herbrand, of *general recursiveness* as an analysis of the most general concept of computability, using systems of equations. However, at the time Gödel regarded this identification only as a "heuristic principle".³ During the same period, Church had conversations with Gödel in which he advanced his own proposal. Gödel regarded this as "thoroughly unsatisfactory" but Church replied that "if [Gödel] would propose any definition of

² There are many variants of the λ -calculus, such as the λ -*K* calculus, the λ -*I* calculus etc.; these will not be distinguished here. The classic presentation of this subject is of course in Church [1941]; an up-to-date and comprehensive exposition is provided by Barendregt [1984].

³ See p. 348 of Gödel [1986].

“effective calculability which seemed even partially satisfactory [Church] would undertake to prove that it was included in lambda-definability” (Kleene [1981], p. 59). It was only as a result of Turing’s later work on mechanical computability that Gödel would accept this identification, albeit indirectly through a chain of equivalences.

The first step in this chain was the demonstration by Church and Kleene that λ -definability for functions of positive integers is equivalent to general recursiveness. Thus when Church finally announced his Thesis in published form [1936], he formulated the Thesis in terms of general recursiveness but bolstered it by the fact of this equivalence (see Davis [1982], pp. 10–11). Church applied his Thesis in [1936] to demonstrate the effective unsolvability of various mathematical and logical problems, including the decision problem for sufficiently strong formal systems. A year later (15 April 1936, just when Turing was presenting the draft of *his* paper to Newman), Church submitted his [1936a] with its simple proof of the unsolvability of the decision problem for the first-order functional calculus of logic.

The dismaying news of Church’s work reached Cambridge in May 1936. At first this seemed to pre-empt Turing’s analysis of computability and his result on the *Entscheidungsproblem*; but Turing’s definition of computability was sufficiently different from that of Church as to warrant separate publication. Thus Turing’s paper was submitted after all, on 28 May 1936; in August 1936 he tacked on an appendix sketching a proof of the equivalence between his notion of computability and λ -definability, thus establishing the second link in the chain of equivalences.

Independently of Turing, but with knowledge of Church’s Thesis, Post also proposed in 1936 a definition of mechanical computability, which was very close to that of Turing. From the conceptual point of view these are now generally regarded as providing the most convincing analyses of the informal notion of effective or mechanical computability. Since 1936 there have been further defined notions of effectiveness, all shown to be equivalent, but there is hardly any other which carries the same immediate conviction that the sought-for general definition is at hand. Indeed, Gödel later championed Turing’s definition in his postscripts to his paper on incompleteness and the 1934 lectures (on the occasion of their reprinting). For example, in the latter he writes 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. ... 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’.”⁴ On the other hand Gödel never remarked in

⁴ Davis [1965], pp. 71–72. Gödel first stated such views publicly in his 1946 Princeton Bicentennial lecture reproduced in Davis [1965], pp. 84–88 and [Gödel 1990].

print concerning his negative views about Church's Thesis as initially formulated.

Church himself agreed early on that Turing's definition was conceptually superior. In his 1937 review⁵ of Turing's paper, after referring to the equivalence proof in the appendix, Church says: "As a matter of fact, there is involved here the equivalence of three different notions: computability by a Turing machine, general recursiveness in the sense of Herbrand–Gödel–Kleene, and λ -definability in the sense of Kleene and the present reviewer. Of these, the first has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately – i.e. without the necessity of proving preliminary theorems. The second and third have the advantage of suitability for embodiment in a system of symbolic logic."

For the actual development of the (abstract) theory of computation, where one must build up a stock of particular functions and establish various closure conditions, both Church's and Turing's definition are equally awkward and unwieldy. In this respect, general recursiveness is superior, and Kleene's particular schemata for that (using primitive recursion and the minimum operator μ) are the most tractable. In Kleene's words: "I myself, perhaps unduly influenced by rather chilly receptions from audiences around 1933–35 to disquisitions on λ -definability, chose, after general recursiveness had appeared, to put my work in that format. . . . I cannot complain about my audiences after 1935. . . ." Still, he defends λ -definability as having ". . . the remarkable feature that it is all contained in a very simple and almost inevitable formulation, arising in a natural connection with no prethought of the result" (Kleene [1981], p. 62).

Paper 2. Computability and λ -definability and Paper 3. Systems of logic based on ordinals

Back to Turing in Cambridge, May 1936: he had been understandably let down by Church's scoop, but gratified that he had independently arrived at the same result concerning the *Entscheidungsproblem*, and that his own work on computability was recognised (if by no one else but Newman and Church) to have independent merit. What next? Newman recommended that he go study with Church in Princeton, and wrote to Church to see if a visit and any sort of grant could be arranged. Turing also applied for a Procter fellowship, offered by Princeton. There were three of these altogether, one designated for Oxford, one for Cambridge, and one for the Collège de France. Turing failed to win the one for Cambridge, and no other money was forthcoming from Princeton. Still, he thought he could just manage on his fellowship funds from Kings College (\$300 p.a.) and decided

⁵ J. Symb. Log. (2) 1937, pp. 42–43.

to spend the year 1936–7 in Princeton; he arrived there at the end of September 1936 (Hodges [1983], pp. 112–117).

Turing was well aware of the growing importance of the Princeton Mathematics Department as a world center for mathematics. It had already been a leader on the American scene when the department was greatly enriched, in the early 1930's, by the establishment of the Institute for Advanced Study (IAS). The two shared the same facilities (Fine Hall) until 1940, so that the lines between them were blurred and there was a great deal of interaction. The IAS had been built up by Oswald Veblen, with Einstein as one of its first members. It benefited particularly as a result of the exodus of mathematicians and physicists from Germany (and especially from the great Mathematical Institute in Göttingen) following the advent of Nazism. Among the mathematical luminaries that Turing found on his arrival in Princeton were Einstein, von Neumann, Weyl, Courant, Hardy and Lefschetz. (Of these, the first three were members of the IAS, Courant and Hardy were visitors, and Lefschetz was in the Princeton department.) Turing wrote home saying that “the mathematics department here comes fully up to expectations”. But in logic, he had hoped to find – besides Church – Gödel, Bernays, Kleene and Rosser “who were here last year (but) have left” (ibid., p. 117). Turing was particularly disappointed to have missed Gödel. As already mentioned, Gödel had visited the IAS in 1933–34; he commenced a second visit in the Fall of 1935 but left after a brief period due to illness. He was to visit the IAS once more, in 1939, before becoming a member in 1940; apparently Gödel and Turing never did meet. Bernays, who had visited in 1935–36, was noted as Hilbert's collaborator on *Beweistheorie*, the proposed means for carrying out Hilbert's foundational program, while Kleene and Rosser (who were, previously, Ph.D. students of Church) had left to take up positions elsewhere. Others whom Turing might have looked to were von Neumann and Weyl; both had taken an active interest in logic in earlier years (of which, see below), but were totally engaged in mainstream mathematics at the time of Turing's arrival on the scene.

Thus Turing was left to draw on Church for direction in the further pursuit of his studies. He dutifully attended Church's lectures “which were rather on the ponderous and laborious side” and took notes on Church's theory of types. Turing also met with him in person from time to time, but Church was “... a retiring man himself, not given to a great deal of discussion.” Turing was himself on the shy side and, indeed, he was described by Max Newman as a “confirmed solitary”. The brief characterisations of Church's style and personality, from Hodges (p. 119), are fair enough, as far as they go. But it should be added that Church was (and is) noted for the great care and precision of his writing and lecturing, and these virtues probably benefited Turing – whose own writing was rough-and-ready and prone to minor errors.

At Church's suggestion, Turing gave a lecture to the Mathematics Club on his work on computability, but the turnout was disappointing. Then in January 1937, when **Paper 1** at last appeared in print, he received only two requests for reprints, one from England and one from Germany. So, for the time being, his work did not attract much attention.

In the period January–April 1937, Turing worked on three relatively brief papers. One (**Paper 2**) was to give more detail about his proof of the equivalence of his notion of computability with that of λ -definability. The other two papers were on group theory; and one of those solved negatively a problem concerning possible approximations of continuous groups, which proved to be of interest to von Neumann. During this period, Dean Eisenhart of the Princeton mathematics Department urged Turing to stay on for a second year and to apply again for a Procter fellowship (worth \$2000 p.a.). He was supported in this by von Neumann, whose letter of recommendation praised Turing for his work on almost periodic functions and continuous groups but made no mention whatever of his work on computability. This time, Turing succeeded in obtaining the fellowship and, having failed to get a position in Cambridge for which he had also applied, decided to stay in Princeton for an extra year and do Ph.D. work under Church.

For the dissertation, it was proposed that Turing take up an idea that had been broached in Church's course, relating to Gödel's incompleteness theorems; presumably this was the kernel of Turing's work on ordinal logics, to be found here as **Paper 3**. After finishing **Paper 2**, he made good progress on **Paper 3** in 1937 and even hoped to finish it by Christmas of that year, but this was not to be: "Church made a number of suggestions which resulted in the thesis being expanded to an appalling length." (Turing, quoted in Hodges [1983], p. 145.) Of course, Church would not knowingly tolerate rough or imprecise formulations and proofs, let alone mistakes – and **Paper 3** shows that Turing went far to meet such demands while still putting his own characteristic stamp on the exposition. The thesis was finally submitted in May 1938 and an oral examination on his work was held at the end of that month. The committee consisted of Church, Lefschetz and Bohnenblust, and Turing's performance was noted as being excellent. The Ph.D. degree itself was granted in June 1938.

Von Neumann thought sufficiently highly of Turing to offer him in May 1938 a position as his assistant at the Institute for Advanced Study, but he turned it down in favour of renewing his fellowship at King's College, which enabled him to return home in July 1938. Beyond publishing his thesis in 1939, Turing was to pursue the topic of ordinal logics no further. The thesis was couched in the formalism of the λ -calculus, very likely at Church's behest, and understanding of the mechanics of the λ -calculus was limited to a rather small audience. This affected its reception for many years.

On computable numbers, with an application to the Entscheidungsproblem

(Proc. Lond. Math. Soc., series 2 vol. **42** (1937), pp. 230–265)

– A correction

(ibid. vol. **43** (1937), p. 544–546)

PREFACE

1. Preamble

This is the classic paper which first established Turing’s reputation and by which he will longest be remembered. The argument falls into three parts.

- A. The notion of a Turing machine is introduced and it is argued that any computation, which can be performed by a human can be imitated by such a machine.
- B. It is shown that there is a universal machine which, when provided with a standard description of any Turing machine will imitate the action of that machine.
- C. A diagonal argument is used to show that there are questions about the actions of Turing machines which cannot be answered by any machine. By formalizing the action of Turing machines in the lower predicate calculus it is shown that the Entscheidungsproblem is mechanically undecidable.

1.1. *History*

Turing always enjoyed calculating. At school he had devised a method for computing π and had used it to calculate the first 36 decimal places [Hodges p. 35]. While walking, bicycling or washing up he would perform mental calculations about mathematical or physical phenomena. Whether calculating mentally or with pencil and paper, Turing was methodical only by fits and starts, and often made mistakes. [When I came to know him later the phrase ‘What’s a factor of two between friends?’ had become a catchword.] But he understood very well what it meant to be totally methodical. Indeed an acceptance – sometimes ready, sometimes reluctant – of the dichotomy between the clearly perceived ideal and the confused actuality was fundamental in Turing’s thought.

In the Spring of 1935 Turing attended lectures by M.H.A. Newman on mathematical logic [Hodges p. 91–93] in which the Entscheidungsproblem (and

Gödel's incompleteness theorem) were discussed. To prove, what by then was commonly though not universally believed, that there could be no effective or mechanical method for deciding which formulae of the predicate calculus are provable, it was necessary to limit the notion of 'effective method' by giving a precise definition of it. The need for such a definition was what immediately stimulated Turing to analyse the process of computation. But it is plain from the paper that he was just as interested in the positive aspects of his analysis as in the negative results, which it enabled him to prove. In particular, section 10 is entirely concerned with what numbers and functions are computable. Ostensibly the purpose of this section is to show that the defined notion of computable has some of the properties which one would expect any intuitive notion to have; but Turing is interested in their properties for their own sake.

I remember Turing telling me that the 'main idea for the paper' came to him when he was lying in the grass in Granchester meadows in the summer of 1935. I assume that he had by then already conceived of some form of Turing machine, and that what he meant by 'the main idea' was the realisation that there could be a universal machine and that this could permit a diagonal argument. Sometime after this he described the universal machine to his friend David Champernowne. He did not discuss his work with Newman, but gave him a completed typewritten draft of the paper in April 1936 [Hodges p. 109]. A little later Newman received from Alonzo Church an off print of his paper [1936] and a pre-print of [1936a], the results of which had been presented to the American Mathematical Society in April 1935. Turing inserted a reference to this in the introduction to his paper, and sent it to the London Mathematical Society where it was received on 8th May. On 31st May Newman wrote to Church:

An off print which you kindly sent me recently of your paper in which you define "calculable numbers" and shew that the Entscheidungsproblem for Hilbert logic is insoluble, had a rather painful interest for a young man, A.M. Turing, here, who was just about to send in for publication a paper in which he had used a definition of "Computable Numbers" for the same purpose.

Church had certainly obtained the result before Turing; but Turing had written his draft without any knowledge of Church's work. It should be remarked that there is no evidence that Turing had read any of the scanty and sporadic literature concerned with the general theory of mechanical computation. In particular, one can be sure that if Turing had read either account by Babbage of the Analytic Engine (Chapter VIII of 'Pages from the life of a philosopher' [1864]) or the account of Menabrea [1842] translated by the Countess of Lovelace [1843] he would have mentioned Babbage's ideas. He might well have read the article on Calculating Machines in the 11th edition of the Encyclopaedia Britannica. In later years he often consulted the copy of the Encyclopaedia which he inherited from

his father – but that article contains only a short and rather dismissive reference to the Analytic Engine: ‘a much more powerful machine... intended to perform any series of possible arithmetical operations’. This would hardly have suggested to Turing that Babbage had in fact conceived a universal machine.

2. Discussion

For an overall account and estimation of the paper, the reader is referred to Newman’s obituary to Turing reproduced in this volume. My paper ‘The confluence of Ideas in 1936’ [Gandy 1988] contains an account of the background history of ideas; and a discussion of the contributions made by Hilbert and his school, by Church and his students and by Post. It also includes a discussion of Turing’s work and its significance. In what follows I shall occasionally draw on that article without indicating the difference between paraphrase and direct quotation.

2.1. *Turing’s analysis of computation*

Turing considers ‘computable (real) numbers’; in fact, what he is considering is total computable functions of a positive integral argument with values 0 and 1. He starts off his detailed analysis (pp. 249–258) by saying:

The real question at issue is “What are the possible processes which can be carried out in computing a real number?”

This is significantly different from the question ‘What is a computable function?’ which other authors asked. Turing, so to speak, pointed himself in the true direction.

2.1.1. He then considers the actions of an abstract human being who is making a calculation; he pictures him as working on squared paper as in “a child’s arithmetic book”. He argues – too briefly – that nothing will be lost by supposing that the calculation is carried out on a potentially infinite tape divided into squares in each of which a single symbol (or none) may be written.

2.1.2. By considering the limitations of our sensory and mental apparatus Turing arrives at the following restrictions on the actions of a computor.⁶

- (1) There is a fixed upper bound to the number of distinct symbols which can be written on a square.
- (2) There is a fixed upper bound on the number of contiguous cells whose contents the computor can take in – ‘at a glance’ as one might say – when he is deciding what to do next.

⁶ I use ‘computor’ for a human being, ‘computer’ for a machine.

(Turing shows by an example that for a normal human being – the reader – this bound, for a linear arrangement, is less than 15. On pp. 250–251 Turing considers the possibility that besides looking at the currently observed squares the computor might also look at some ‘immediately recognised’ specially marked squares. But there must be a fixed upper bound on the number of immediately recognised squares, and so – without detailed argument – he claims that they can be, in effect, adjoined to the observed squares.)

(3) At each step the computor may alter the contents of only one square, and there is a fixed upper bound to the distance the computor can move to reach this square from the observed squares; so we may suppose that it is one of them.

(4) There is a fixed upper bound to the distance between the squares observed at one stage and those observed at the next stage.

(5) There is a fixed upper bound to the number of ‘states of mind’ of the computor: his ‘state of mind’, together with the contents of the observed squares, uniquely determine the action he takes (printing and moving his field of observation), and his next ‘state of mind’. In place of ‘state of mind’ Turing admits that the computor might leave an instruction on how to continue (p. 253).

2.1.3. Thus the computor must follow a fixed, finite, totally explicit set of instructions satisfying the above restrictions. It is then easy to see that his action can be simulated by a Turing machine – which at each step observes a single square, can alter only the contents of that square, and moves by at most one square.

2.1.4. It is worth emphasising that Turing’s analysis is quite explicitly concerned with calculations performed by a human being; there is no reference to machines other than those which he introduces to imitate the actions of a human computor. In subsequent discussions of Turing’s work this fact has sometimes been obscured by a play on the uses of the word ‘mechanical’, which has often since the seventeenth century been applied in a loose figurative sense to human actions. [The earliest example (1607) given in the Oxford English Dictionary refers to farriers who mistreat sick horses by rule of thumb]. Of course, it is not surprising that Turing does not mention machines. Numerical calculation in 1936 was carried out by human beings; they used mechanical aids for performing standard arithmetical operations, but these aids were not programmable. According to Randall ([1982], p. 160) the first general purpose programme-controlled machine to be built and used was Zuse’s machine completed in 1941; even this (and other machines of the same generation) did not fully allow for conditional branching.

Indeed, Turing’s analysis does not directly apply to (discretely acting) machines, since it takes no account of the possibilities of parallel action. If one considers (as in Newtonian theory) the possibility of instantaneous action at a

distance, then the alteration of the record need not be local nor locally determined. However, if (as in the theory of relativity) one supposes that there is an upper bound to the velocity of propagation of physical influences, then one can establish Turing's thesis for machines as well as for human beings (see Gandy [1981]).

2.1.5. What makes Turing's analysis so breathtaking is its combination of generality, directness and simplicity. These are characteristic of his way of thinking; but the particular manner in which they come to the fore in this paper is a result of his having asked himself the question 'What is a computable real number?' rather than, say, 'What functions are computable?'. The latter question leads most naturally to a consideration of the various different ways in which calculable functions may be specified; thus Hilbert and his school investigated various kinds of recursive definition (see Hilbert [1926] and Ackermann [1928]). Had Turing started with this line of thought, his machines might have been described in terms of a high-level language (such as LISP, say) rather than the extremely simple machine – language which he actually uses. But then the argument that all conceivable methods of calculation had been covered would have been far less direct and less cogent.

2.1.6. Turing concentrated on implementation rather than specification, and found that this made it possible to set an exact limit on what is calculable. He did, however, use the notion of specification as a supporting argument – for predicates in §9.II, and for functions in the first paragraph of §10. A predicate G is specified by giving a formula \mathbf{A} (of first-order predicate calculus using only relational symbols) which implicitly defines it together with the requirement that G be *formally reckonable* in the predicate calculus; i.e. that the appropriate formula $G(x)$ or $\neg G(x)$ can be inferred from \mathbf{A} and the formula which expresses that x is the n -th natural number. One can say that the specification \mathbf{A} of G must be implemented by formal proofs. But this supporting argument lacks the generality of Turing's direct proof of his thesis. Firstly, recursions involving higher type objects may be used in specifying calculable functions; see, for example, p. 389 of Hilbert [1926]. Secondly, one might wish to interpret 'formal reckonability' as allowing proofs in some formal system more powerful than the predicate calculus; this possibility is considered by Church ([1936], p. 357) at the corresponding point in *his* argument.

2.2. *Philosophical significance of Turing's analysis*

2.2.1. Formal systems. Turing's work makes it possible to give a satisfactory and definitive characterisation of a formal system (or theory). Namely, a formal

system is one whose expressions are built up from a finite list of primitive symbols and for which there is a Turing machine which will test all its theorems (or, more generally, all its inferences). Turing does not refer to that possibility in this paper, but the characterisation is of fundamental importance for the philosophy of mathematics. It allows Gödel's proof of incompleteness to be applied to any formal system, which contains a certain amount of elementary number theory, and was much emphasised at times by Gödel in 1963–1965 (*Volume 2 of his Collected Works* [1990]). Hence, it sets limits to what can be rigorously proved in any formalisation of mathematics. In his paper on ordinal logics, which appears later in this volume, Turing gives a model for the different roles, which are played in mathematics by intuition and by formal proofs. This notion allows one to generalise the notions of formal specification and formal reckonability described in 2.1.6 above; but one cannot, of course, use the fact that they give an equivalent definition of computability as a supporting argument for Turing's thesis – this would result in a vicious circle.

Note: It should be mentioned that in the early 1920's Post had developed a quite different (but eventually seen to be equivalent) characterisation of formal system. This was not published until Post [1943]; for an historical account see Post [1965].

2.2.2. Wittgenstein's paradox. Turing's analysis can be applied quite generally to characterise the notion of a *rule* – of calculation, of inference, of procedure, of construction – and so on. In mathematics, such rules are usually designed so that they may be applied in a potentially infinite number of distinct situations. This gives rise to a puzzle or paradox with which Wittgenstein was much concerned. Namely, suppose one has witnessed someone (perhaps oneself) apply a given rule in a finite number of cases; how can one be sure that one has applied the given rule and not some other rule which agrees with it so far? Hence, how can one ever prove that a particular rule has been correctly applied? The paradox is not immediately resolved by requiring that the particular rule be clearly stated; for the statement cannot list what is to be done in all the infinite number of situations in which the rule may be applied.

Turing's analysis does not solve this paradox, but it does make clearer where the heart of the problem lies. I will call a finite list of actions which are to be taken in suitable circumstances a *recipe*; such, for example, is the set of instructions which one might give out for travelling from one place to another. A recipe may well include alternatives. The table of instructions (or programme) for a Turing machine constitutes a recipe. What Turing showed was that applying a rule in a given situation can always be reduced to the iterated application of a recipe. Thus the essential puzzle is: 'How can one prove that someone has followed a recipe correctly?'; or 'How can one prove that a situation (or even an action) is the

same on one occasion as it was on another?’ Wittgenstein certainly recognised this form of the puzzle; he discusses, for example, how one can *know* that a colour word has been correctly used. His solution draws attention to the fact that correctness is to be judged by the ability to communicate with others, and so cannot be applied to the behaviour of a totally isolated individual. But he sometimes talked as if the puzzle about rules (and proofs) was different from, was not merely a rhetorical embellishment of, the puzzle about recipes. In particular, so it seems to me, he did so in the lectures on the foundations of mathematics (recorded in Wittgenstein [1976]) which he gave in 1939 and at which Turing was a vocal participant. I find it surprising that Turing seems never to have made the point which I have just discussed. A full and clear account of Wittgenstein’s paradox and his solution of it will be found in Kripke [1982].

2.3. Turing machines and electronic computers

2.3.1. The influential ideas. The title of Turing’s paper and the fact that its first section is headed ‘computing machines’ encouraged people concerned with the design of computers to read it, or at least to look at it; but, the ideas it contains would have been equally important if Turing, like Post, had avoided the use of the word ‘machine’. I think that these ideas, in order of importance, are as follows:

- (i) The elementary steps are extremely simple, and have specifications of a fixed length.
- (ii) The universal machine is a stored-program machine; that is, unlike Babbage’s all-purpose machine, the mechanisms used in reading a program are of the same kind as those used in executing it.
- (iii) Conditional instructions are no different from unconditional ones.
- (iv) The operation is easily adapted to binary storage and working.

Notes: (1) Turing presumably realised when he wrote the paper that, like Post, he could have used ‘mark’ and ‘blank’ as the only symbols; but that had he done so his table for the universal machine would have been totally unreadable.

(2) Kleene used binary Turing machines in his book [1952].

It is (i) and (iv) that made it possible for McCulloch and Pitts to show that the control mechanisms of a Turing machine can be simulated by a finite network of ‘neurons’ (gates with delays). The λ -calculus and the equational calculus both use ‘substitute a given term for a given variable in a given expression’ as an elementary step; this cannot have a total specification of fixed length. On the other hand, the λ -calculus is a stored program device, since there is no difference between a program (a λ -term) and the successive stages in its computation. Although the earlier designs for computers (in particular, the EDVAC) only allow rather restricted use of conditional instructions, the use of gates with two or

more inputs does in fact reflect the conditional nature of the elementary steps of a Turing machine.

2.3.2. Their effect on U.S. developments. There is some controversy about the exact extent of the influence of Turing's ideas on the design of electronic computers in the U.S.A. I record a few historical facts:

(1) By the late 1930's von Neumann had become familiar with Turing's ideas and was enthusiastic about them (Randell [1972], p. 10; Hodges [1983], p. 145; Davis [1987]). There is no evidence, however, that there was ever a meeting at which they exchanged ideas about the possibilities of designing 'universal' electronic computers. They did meet early in 1947, but by that time they had written their respective reports. In a perceptive footnote ([1983], fn 5.26 on pp. 555–556) Hodges argues that such a meeting would not, in any case, have been likely to have been of great importance; each would develop his own ideas.

(2) In 1945 von Neumann wrote his 'First draft of a report on the EDVAC'. In this he makes considerable use of the idealised neuronal networks of McCulloch and Pitts [1943]. He does not explicitly refer to Turing [1936–7] in that paper (although in his Hixon lecture in 1948 [1951] he gives Turing his due). As with Turing's universal machine, the program is stored in a special part of the memory. This report circulated widely and was influential. Turing read it in the summer of 1945.

(3) Most of the essential ideas for the design of electronic computers – binary working, use of logical circuits and stored programs – were developed independently by various people, from 1936 onwards; see Randell [1982], chapters VII and VIII, and Burks [1980]. The most important and influential place for the construction of electronic computers in the 1940's was the Moore School of Engineering; the first working large-scale (18,000 valves) electronic computer was the ENIAC. The original proposal (in April 1943) was by Mauchly and Eckert. The machine was finished at the end of 1945. In the meantime, plans were being made for 'EDVAC-type' machines; von Neumann became a consultant there in September 1944. Much of the work on ENIAC and EDVAC was classified. It has become, it seems, impossible to discern a linear flow of ideas; probably there was no such thing. Much acrimony (see Eckert [1980] and Mauchly [1980]) and a protracted legal battle developed from the question of who told what to whom.

(4) According to Randell [1980], Turing and Newman and a group of mathematicians and engineers at Bletchley Park discussed in 1942–43, out of office hours, Turing's universal machine, Babbage's plans for the Analytic Engine, and the possibilities of artificial intelligence. Turing played some part in the design of the first machines built under Newman's direction. The final machines ('Colossi') can claim to be the first medium-sized (2,000–3,000 valves) fairly flexible, pro-

grammable electronic computers. (The Mark II Colossus came into service in June 1944.)

(5) Turing worked at the National Physical Laboratory from 1945 to 1948 designing a computer. His report on the ACE [1945] was submitted in March 1946. Turing was influenced by von Neumann [1945], but he describes, in fair detail, the design of a quite specific machine, and, ahead of the times, proposed that many of the operations of the machine should have been effected by writing subroutines, rather than by building special single-purpose units. In his lecture [1947] to the London Mathematical Society he traces the connections between the design of the ACE and his 1936–7 paper.

The ideas of the ACE report are discussed at length by Hodges [1983], pp. 317–333.

(6) At the same time, in Manchester, Newman was responsible for, and contributed to, the design of what became the Manchester Mark 1 machine (Newman [1948]). Turing joined him in the autumn of 1948. His chief interest was in using the machine (and helping others to use it). Although he wrote the ‘Programmer’s Handbook’ he did not contribute to the design of the machine or of programming languages for it. Hodges [1983, p. 401] lists some of the things which Turing could have worked on but didn’t.

2.3.3. Summary. Although it may be difficult to trace the precise influence of this paper on the design and development of high-speed digital computers, its fundamental importance for the theory of computation is clear. Turing machines (and modifications of them) still provide the standard setting for the definition of the complexity of computation in terms of bounds on time and space; together with the neural nets of McCulloch and Pitts they provided the foundations of the theory of automata; together with the generated sets of Post [1943] they provided the foundation for the theory of formal grammars.

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO
THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The “computable” numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbersome technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π , e , etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, “Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I”, *Monatshefte Math. Phys.*, 38 (1931), 173–198.

have valuable applications. In particular, it is shown (§11) that the Hilbertian Entscheidungsproblem can have no solution.

In a recent paper Alonzo Church† has introduced an idea of “effective calculability”, which is equivalent to my “computability”, but is very differently defined. Church also reaches similar conclusions about the Entscheidungsproblem‡. The proof of equivalence between “computability” and “effective calculability” is outlined in an appendix to the present paper.

1. Computing machines.

We have said that the computable numbers are those whose decimals are calculable by finite means. This requires rather more explicit definition. No real attempt will be made to justify the definitions given until we reach §9. For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited.

We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions q_1, q_2, \dots, q_R which will be called “ m -configurations”. The machine is supplied with a “tape” (the analogue of paper) running through it, and divided into sections (called “squares”) each capable of bearing a “symbol”. At any moment there is just one square, say the r -th, bearing the symbol $\mathfrak{S}(r)$ which is “in the machine”. We may call this square the “scanned square”. The symbol on the scanned square may be called the “scanned symbol”. The “scanned symbol” is the only one of which the machine is, so to speak, “directly aware”. However, by altering its m -configuration the machine can effectively remember some of the symbols which it has “seen” (scanned) previously. The possible behaviour of the machine at any moment is determined by the m -configuration q_n and the scanned symbol $\mathfrak{S}(r)$. This pair $q_n, \mathfrak{S}(r)$ will be called the “configuration”: thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (*i.e.* bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the m -configuration may be changed. Some of the symbols written down

† Alonzo Church, “An unsolvable problem of elementary number theory”, *American J. of Math.*, 58 (1936), 345–363.

‡ Alonzo Church, “A note on the Entscheidungsproblem”, *J. of Symbolic Logic*, 1 (1936), 40–41.

will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to “assist the memory”. It will only be these rough notes which will be liable to erasure.

It is my contention that these operations include all those which are used in the computation of a number. The defence of this contention will be easier when the theory of the machines is familiar to the reader. In the next section I therefore proceed with the development of the theory and assume that it is understood what is meant by “machine”, “tape”, “scanned”, etc.

2. *Definitions.*

Automatic machines.

If at each stage the motion of a machine (in the sense of §1) is *completely* determined by the configuration, we shall call the machine an “automatic machine” (or *a*-machine).

For some purposes we might use machines (choice machines or *c*-machines) whose motion is only partially determined by the configuration (hence the use of the word “possible” in §1). When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. This would be the case if we were using machines to deal with axiomatic systems. In this paper I deal only with automatic machines, and will therefore often omit the prefix *a*-.

Computing machines.

If an *a*-machine prints two kinds of symbols, of which the first kind (called figures) consists entirely of 0 and 1 (the others being called symbols of the second kind), then the machine will be called a computing machine. If the machine is supplied with a blank tape and set in motion, starting from the correct initial *m*-configuration, the subsequence of the symbols printed by it which are of the first kind will be called the *sequence computed by the machine*. The real number whose expression as a binary decimal is obtained by prefacing this sequence by a decimal point is called the *number computed by the machine*.

At any stage of the motion of the machine, the number of the scanned square, the complete sequence of all symbols on the tape, and the *m*-configuration will be said to describe the *complete configuration* at that stage. The changes of the machine and tape between successive complete configurations will be called the *moves* of the machine.

Circular and circle-free machines.

If a computing machine never writes down more than a finite number of symbols of the first kind, it will be called *circular*. Otherwise it is said to be *circle-free*.

A machine will be circular if it reaches a configuration from which there is no possible move, or if it goes on moving, and possibly printing symbols of the second kind, but cannot print any more symbols of the first kind. The significance of the term “circular” will be explained in § 8.

Computable sequences and numbers.

A sequence is said to be computable if it can be computed by a circle-free machine. A number is computable if it differs by an integer from the number computed by a circle-free machine.

We shall avoid confusion by speaking more often of computable sequences than of computable numbers.

3. Examples of computing machines.

I. A machine can be constructed to compute the sequence 010101.... The machine is to have the four m -configurations “ b ”, “ c ”, “ f ”, “ e ” and is capable of printing “0” and “1”. The behaviour of the machine is described in the following table in which “ R ” means “the machine moves so that it scans the square immediately on the right of the one it was scanning previously”. Similarly for “ L ”. “ E ” means “the scanned symbol is erased” and “ P ” stands for “prints”. This table (and all succeeding tables of the same kind) is to be understood to mean that for a configuration described in the first two columns the operations in the third column are carried out successively, and the machine then goes over into the m -configuration described in the last column. When the second column is left blank, it is understood that the behaviour of the third and fourth columns applies for any symbol and for no symbol. The machine starts in the m -configuration b with a blank tape.

<i>Configuration</i>		<i>Behaviour</i>		
<i>m-config.</i>	<i>symbol</i>	<i>operations</i>	<i>final</i>	<i>m-config.</i>
b	None	$P0, R$		c
c	None	R		c
e	None	$P1, R$		f
f	None	R		e

If (contrary to the description in § 1) we allow the letters L , R to appear more than once in the operations column we can simplify the table considerably.

<i>m-config.</i>	<i>symbol</i>	<i>operations</i>	<i>final m-config.</i>
\mathfrak{b}	None	$P0$	\mathfrak{b}
	0	$R, R, P1$	\mathfrak{b}
	1	$R, R, P0$	\mathfrak{b}

II. As a slightly more difficult example we can construct a machine to compute the sequence $001011011101111011111\dots$. The machine is to be capable of five *m-configurations*, viz. “ \mathfrak{o} ”, “ \mathfrak{q} ”, “ \mathfrak{p} ”, “ \mathfrak{f} ”, “ \mathfrak{b} ” and of printing “ \mathfrak{o} ”, “ x ”, “0”, “1”. The first three symbols on the tape will be “ $\mathfrak{o}\mathfrak{o}0$ ”; the other figures follow on alternate squares. On the intermediate squares we never print anything but “ x ”. These letters serve to “keep the place” for us and are erased when we have finished with them. We also arrange that in the sequence of figures on alternate squares there shall be no blanks.

<i>Configuration</i>		<i>Behaviour</i>	
<i>m-config.</i>	<i>symbol</i>	<i>operations</i>	<i>final m-config.</i>
\mathfrak{b}		$P\mathfrak{o}, R, P\mathfrak{o}, R, P0, R, R, P0, L, L$	\mathfrak{o}
\mathfrak{o}	1	R, Px, L, L, L	\mathfrak{o}
	0		\mathfrak{q}
\mathfrak{q}	Any (0 or 1)	R, R	\mathfrak{q}
	None	$P1, L$	\mathfrak{p}
\mathfrak{p}	x	E, R	\mathfrak{q}
	\mathfrak{o}	R	\mathfrak{f}
	None	L, L	\mathfrak{p}
\mathfrak{f}	Any	R, R	\mathfrak{f}
	None	$P0, L, L$	\mathfrak{o}

To illustrate the working of this machine a table is given below of the first few complete configurations. These complete configurations are described by writing down the sequence of symbols which are on the tape,

with the m -configuration written below the scanned symbol. The successive complete configurations are separated by colons.

: e e 0	0	:	e e 0	0	:	e e 0	0	:	e e 0	0	1	:
b	v		q			q			q		p	
e e 0	0	1	:	e e 0	0	1	:	e e 0	0	1	:	e e 0
p			p			f			f		f	
e e 0	0	1	:	e e 0	0	1	:	e e 0	0	1	0	:
			f			f			v			
e e 0	0	1	x	0	:						
			q									

This table could also be written in the form

$$b : e e v 0 \quad 0 : e e q 0 \quad 0 : \dots, \quad (C)$$

in which a space has been made on the left of the scanned symbol and the m -configuration written in this space. This form is less easy to follow, but we shall make use of it later for theoretical purposes.

The convention of writing the figures only on alternate squares is very useful: I shall always make use of it. I shall call the one sequence of alternate squares F -squares and the other sequence E -squares. The symbols on E -squares will be liable to erasure. The symbols on F -squares form a continuous sequence. There are no blanks until the end is reached. There is no need to have more than one E -square between each pair of F -squares: an apparent need of more E -squares can be satisfied by having a sufficiently rich variety of symbols capable of being printed on E -squares. If a symbol β is on an F -square S and a symbol α is on the E -square next on the right of S , then S and β will be said to be *marked* with α . The process of printing this α will be called marking β (or S) with α .

4. Abbreviated tables.

There are certain types of process used by nearly all machines, and these, in some machines, are used in many connections. These processes include copying down sequences of symbols, comparing sequences, erasing all symbols of a given form, etc. Where such processes are concerned we can abbreviate the tables for the m -configurations considerably by the use of "skeleton tables". In skeleton tables there appear capital German letters and small Greek letters. These are of the nature of "variables". By replacing each capital German letter throughout by an m -configuration

and each small Greek letter by a symbol, we obtain the table for an m -configuration.

The skeleton tables are to be regarded as nothing but abbreviations: they are not essential. So long as the reader understands how to obtain the complete tables from the skeleton tables, there is no need to give any exact definitions in this connection.

Let us consider an example:

m -config. Symbol Behaviour Final
 m -config.

$f(\mathfrak{C}, \mathfrak{B}, a)$	$\begin{cases} \mathfrak{a} \\ \text{not } \mathfrak{a} \end{cases}$	L	$f_1(\mathfrak{C}, \mathfrak{B}, a)$	From the m -configuration $f(\mathfrak{C}, \mathfrak{B}, a)$ the machine finds the symbol of form a which is farthest to the left (the "first a ") and the m -configuration then becomes \mathfrak{C} . If there is no a then the m -configuration becomes \mathfrak{B} .
$f_1(\mathfrak{C}, \mathfrak{B}, a)$	$\begin{cases} a \\ \text{not } a \\ \text{None} \end{cases}$	R	$f_1(\mathfrak{C}, \mathfrak{B}, a)$	
$f_2(\mathfrak{C}, \mathfrak{B}, a)$	$\begin{cases} a \\ \text{not } a \\ \text{None} \end{cases}$	R	$f_1(\mathfrak{C}, \mathfrak{B}, a)$	

If we were to replace \mathfrak{C} throughout by q (say), \mathfrak{B} by r , and a by x , we should have a complete table for the m -configuration $f(q, r, x)$. f is called an " m -configuration function" or " m -function".

The only expressions which are admissible for substitution in an m -function are the m -configurations and symbols of the machine. These have to be enumerated more or less explicitly: they may include expressions such as $p(\epsilon, x)$; indeed they must if there are any m -functions used at all. If we did not insist on this explicit enumeration, but simply stated that the machine had certain m -configurations (enumerated) and all m -configurations obtainable by substitution of m -configurations in certain m -functions, we should usually get an infinity of m -configurations; *e.g.*, we might say that the machine was to have the m -configuration q and all m -configurations obtainable by substituting an m -configuration for \mathfrak{C} in $p(\mathfrak{C})$. Then it would have $q, p(q), p(p(q)), p(p(p(q))), \dots$ as m -configurations.

Our interpretation rule then is this. We are given the names of the m -configurations of the machine, mostly expressed in terms of m -functions. We are also given skeleton tables. All we want is the complete table for the m -configurations of the machine. This is obtained by repeated substitution in the skeleton tables.

Further examples.

(In the explanations the symbol “ \rightarrow ” is used to signify “the machine goes into the m -configuration. . . .”)

$e(\mathfrak{C}, \mathfrak{B}, a)$ $f(e_1(\mathfrak{C}, \mathfrak{B}, a), \mathfrak{B}, a)$ From $e(\mathfrak{C}, \mathfrak{B}, a)$ the first a is
 $e_1(\mathfrak{C}, \mathfrak{B}, a)$ E \mathfrak{C} erased and $\rightarrow \mathfrak{C}$. If there is no
 $a \rightarrow \mathfrak{B}$.

$e(\mathfrak{B}, a)$ $e(e(\mathfrak{B}, a), \mathfrak{B}, a)$ From $e(\mathfrak{B}, a)$ all letters a are
 $e_1(\mathfrak{B}, a)$ E \mathfrak{B} erased and $\rightarrow \mathfrak{B}$.

The last example seems somewhat more difficult to interpret than most. Let us suppose that in the list of m -configurations of some machine there appears $e(b, x)$ ($= q$, say). The table is

$e(b, x)$	$f(e(b, x), b, x)$	
or	q	$e(q, b, x)$.

Or, in greater detail:

q		$e(q, b, x)$
$e(q, b, x)$	$f(e_1(q, b, x), b, x)$	
$e_1(q, b, x)$	E	q .

In this we could replace $e_1(q, b, x)$ by q' and then give the table for f (with the right substitutions) and eventually reach a table in which no m -functions appeared.

$pe(\mathfrak{C}, \beta)$	$f(pe_1(\mathfrak{C}, \beta), \mathfrak{C}, \beta)$	
$pe_1(\mathfrak{C}, \beta)$	$\begin{cases} \text{Any} & R, R \\ \text{None} & P\beta \end{cases}$	$pe_1(\mathfrak{C}, \beta)$ From $pe(\mathfrak{C}, \beta)$ the machine \mathfrak{C} prints β at the end of the \mathfrak{C} sequence of symbols and $\rightarrow \mathfrak{C}$.
$l(\mathfrak{C})$	L	\mathfrak{C} From $f'(\mathfrak{C}, \mathfrak{B}, a)$ it does the
$r(\mathfrak{C})$	R	\mathfrak{C} same as for $f(\mathfrak{C}, \mathfrak{B}, a)$ but \mathfrak{C} moves to the left before $\rightarrow \mathfrak{C}$.
$f'(\mathfrak{C}, \mathfrak{B}, a)$	$f(l(\mathfrak{C}), \mathfrak{B}, a)$	
$f''(\mathfrak{C}, \mathfrak{B}, a)$	$f(r(\mathfrak{C}), \mathfrak{B}, a)$	
$e(\mathfrak{C}, \mathfrak{B}, a)$	$f'(e_1(\mathfrak{C}), \mathfrak{B}, a)$	$e(\mathfrak{C}, \mathfrak{B}, a)$. The machine
$e_1(\mathfrak{C})$	β	$pe(\mathfrak{C}, \beta)$ writes at the end the first sym- \mathfrak{C} β and $\rightarrow \mathfrak{C}$.

The last line stands for the totality of lines obtainable from it by replacing β by any symbol which may occur on the tape of the machine concerned.

$ce(\mathfrak{C}, \mathfrak{B}, a)$	$c(e(\mathfrak{C}, \mathfrak{B}, a), \mathfrak{B}, a)$	$ce(\mathfrak{B}, a)$. The machine copies down in order at the end all symbols marked a and erases the letters a ; $\rightarrow \mathfrak{B}$.
$ce(\mathfrak{B}, a)$	$ce(ce(\mathfrak{B}, a), \mathfrak{B}, a)$	
$re(\mathfrak{C}, \mathfrak{B}, a, \beta)$	$f(re_1(\mathfrak{C}, \mathfrak{B}, a, \beta), \mathfrak{B}, a)$	$re(\mathfrak{C}, \mathfrak{B}, a, \beta)$. The machine replaces the first a by β and $\rightarrow \mathfrak{C} \rightarrow \mathfrak{B}$ if there is no a .
$re_1(\mathfrak{C}, \mathfrak{B}, a, \beta)$	$E, P\beta$	
$re(\mathfrak{B}, a, \beta)$	E	$re(\mathfrak{B}, a, \beta)$. The machine replaces all letters a by β ; $\rightarrow \mathfrak{B}$.
$cr(\mathfrak{C}, \mathfrak{B}, a)$	$c(re(\mathfrak{C}, \mathfrak{B}, a, a), \mathfrak{B}, a)$	$cr(\mathfrak{B}, a)$ differs from $ce(\mathfrak{B}, a)$ only in that the letters a are not erased. The m -configuration $cr(\mathfrak{B}, a)$ is taken up when no letters "a" are on the tape.
$cr(\mathfrak{B}, a)$	$cr(cr(\mathfrak{B}, a), re(\mathfrak{B}, a, a), a)$	
$cp(\mathfrak{C}, \mathfrak{U}, \mathfrak{E}, a, \beta)$	$f'(cp_1(\mathfrak{C}, \mathfrak{U}, \beta), f(\mathfrak{U}, \mathfrak{E}, \beta), a)$	
$cp_1(\mathfrak{C}, \mathfrak{U}, \beta)$	γ	$f'(cp_2(\mathfrak{C}, \mathfrak{U}, \gamma), \mathfrak{U}, \beta)$
$cp_2(\mathfrak{C}, \mathfrak{U}, \gamma)$	$\begin{cases} \gamma & \mathfrak{C} \\ \text{not } \gamma & \mathfrak{U}. \end{cases}$	

The first symbol marked a and the first marked β are compared. If there is neither a nor β , $\rightarrow \mathfrak{E}$. If there are both and the symbols are alike, $\rightarrow \mathfrak{C}$. Otherwise $\rightarrow \mathfrak{U}$.

$$cpe(\mathfrak{C}, \mathfrak{U}, \mathfrak{E}, a, \beta) \quad cp \left(e \left(e(\mathfrak{C}, \mathfrak{E}, \beta), \mathfrak{C}, a \right), \mathfrak{U}, \mathfrak{E}, a, \beta \right)$$

$cpe(\mathfrak{C}, \mathfrak{U}, \mathfrak{E}, a, \beta)$ differs from $cp(\mathfrak{C}, \mathfrak{U}, \mathfrak{E}, a, \beta)$ in that in the case when there is similarity the first a and β are erased.

$$cpe(\mathfrak{U}, \mathfrak{E}, a, \beta) \quad cpe \left(cpe(\mathfrak{U}, \mathfrak{E}, a, \beta), \mathfrak{U}, \mathfrak{E}, a, \beta \right).$$

$cpe(\mathfrak{U}, \mathfrak{E}, a, \beta)$. The sequence of symbols marked a is compared with the sequence marked β . $\rightarrow \mathfrak{E}$ if they are similar. Otherwise $\rightarrow \mathfrak{U}$. Some of the symbols a and β are erased.

$q(\mathfrak{C})$	$\begin{cases} \text{Any} & R \\ \text{None} & R \end{cases}$	$q(\mathfrak{C})$	$q(\mathfrak{C}, a)$. The machine finds the last symbol of form a . $\rightarrow \mathfrak{C}$.
$q_1(\mathfrak{C})$	$\begin{cases} \text{Any} & R \\ \text{None} & \mathfrak{C} \end{cases}$	$q(\mathfrak{C})$	
$q(\mathfrak{C}, a)$		$q(q_1(\mathfrak{C}, a))$	
$q_1(\mathfrak{C}, a)$	$\begin{cases} a & \mathfrak{C} \\ \text{not } a & L \end{cases}$	$q_1(\mathfrak{C}, a)$	
$pe_2(\mathfrak{B}, a, \beta)$		$pe(pe(\mathfrak{B}, \beta), a)$	$pe_2(\mathfrak{B}, a, \beta)$. The machine prints $a \beta$ at the end.
$ce_2(\mathfrak{B}, a, \beta)$		$ce(ce(\mathfrak{B}, \beta), a)$	
$ce_3(\mathfrak{B}, a, \beta, \gamma)$		$ce(ce_2(\mathfrak{B}, \beta, \gamma), a)$	$ce_3(\mathfrak{B}, a, \beta, \gamma)$. The machine copies down at the end first the symbols marked a , then those marked β , and finally those marked γ ; it erases the symbols a, β, γ .
$e(\mathfrak{C})$	$\begin{cases} \text{o} & R \\ \text{Not o} & L \end{cases}$	$e_1(\mathfrak{C})$	From $e(\mathfrak{C})$ the marks are erased from all marked symbols. $\rightarrow \mathfrak{C}$.
$e_1(\mathfrak{C})$	$\begin{cases} \text{Any} & R, E, R \\ \text{None} & \mathfrak{C} \end{cases}$	$e_1(\mathfrak{C})$	

5. Enumeration of computable sequences.

A computable sequence γ is determined by a description of a machine which computes γ . Thus the sequence 001011011101111... is determined by the table on p. 234, and, in fact, any computable sequence is capable of being described in terms of such a table.

It will be useful to put these tables into a kind of standard form. In the first place let us suppose that the table is given in the same form as the first table, for example, I on p. 233. That is to say, that the entry in the operations column is always of one of the forms $E : E$, $R : E$, $L : Pa : Pa$, $R : Pa$, $L : R : L$: or no entry at all. The table can always be put into this form by introducing more m -configurations. Now let us give numbers to the m -configurations, calling them q_1, \dots, q_R , as in §1. The initial m -configuration is always to be called q_1 . We also give numbers to the symbols S_1, \dots, S_m

and, in particular, blank = S_0 , $0 = S_1$, $1 = S_2$. The lines of the table are now of form

<i>m-config.</i>	<i>Symbol</i>	<i>Operations</i>	<i>Final</i> <i>m-config.</i>
q_i	S_j	PS_k, L	q_m (N_1)
q_i	S_j	PS_k, R	q_m (N_2)
q_i	S_j	PS_k	q_m (N_3)

Lines such as

q_i	S_j	E, R	q_m
-------	-------	--------	-------

are to be written as

q_i	S_j	PS_0, R	q_m
-------	-------	-----------	-------

and lines such as

q_i	S_j	R	q_m
-------	-------	-----	-------

to be written as

q_i	S_j	PS_j, R	q_m
-------	-------	-----------	-------

In this way we reduce each line of the table to a line of one of the forms (N_1) , (N_2) , (N_3) .

From each line of form (N_1) let us form an expression $q_i S_j S_k L q_m$; from each line of form (N_2) we form an expression $q_i S_j S_k R q_m$; and from each line of form (N_3) we form an expression $q_i S_j S_k N q_m$.

Let us write down all expressions so formed from the table for the machine and separate them by semi-colons. In this way we obtain a complete description of the machine. In this description we shall replace q_i by the letter "D" followed by the letter "A" repeated i times, and S_j by "D" followed by "C" repeated j times. This new description of the machine may be called the *standard description* (S.D.). It is made up entirely from the letters "A", "C", "D", "L", "R", "N", and from ";;".

If finally we replace "A" by "1", "C" by "2", "D" by "3", "L" by "4", "R" by "5", "N" by "6", and ";" by "7" we shall have a description of the machine in the form of an arabic numeral. The integer represented by this numeral may be called a *description number* (D.N) of the machine. The D.N determine the S.D and the structure of the

machine uniquely. The machine whose D.N is n may be described as $\mathcal{M}(n)$.

To each computable sequence there corresponds at least one description number, while to no description number does there correspond more than one computable sequence. The computable sequences and numbers are therefore enumerable.

Let us find a description number for the machine I of §3. When we rename the m -configurations its table becomes:

q_1	S_0	PS_1, R	q_2
q_2	S_0	PS_0, R	q_3
q_3	S_0	PS_2, R	q_4
q_4	S_0	PS_0, R	q_1

Other tables could be obtained by adding irrelevant lines such as

q_1	S_1	PS_1, R	q_2
-------	-------	-----------	-------

Our first standard form would be

$$q_1 S_0 S_1 R q_2; q_2 S_0 S_0 R q_3; q_3 S_0 S_2 R q_4; q_4 S_0 S_0 R q_1;.$$

The standard description is

DADDCCRDAA;DAADDRDAAA;

DAAADDCCRDAAAA;DAAAADDRDA;

A description number is

31332531173113353111731113322531111731111335317

and so is

3133253117311335311173111332253111173111133531731323253117

A number which is a description number of a circle-free machine will be called a *satisfactory* number. In §8 it is shown that there can be no general process for determining whether a given number is satisfactory or not.

6. The universal computing machine.

It is possible to invent a single machine which can be used to compute any computable sequence. If this machine \mathcal{U} is supplied with a tape on the beginning of which is written the S.D of some computing machine \mathcal{M} ,

then \mathcal{U} will compute the same sequence as \mathcal{M} . In this section I explain in outline the behaviour of the machine. The next section is devoted to giving the complete table for \mathcal{U} .

Let us first suppose that we have a machine \mathcal{M}' which will write down on the F -squares the successive complete configurations of \mathcal{M} . These might be expressed in the same form as on p. 235, using the second description, (C), with all symbols on one line. Or, better, we could transform this description (as in § 5) by replacing each m -configuration by “ D ” followed by “ A ” repeated the appropriate number of times, and by replacing each symbol by “ D ” followed by “ C ” repeated the appropriate number of times. The numbers of letters “ A ” and “ C ” are to agree with the numbers chosen in § 5, so that, in particular, “0” is replaced by “ DC ”, “1” by “ DCC ”, and the blanks by “ D ”. These substitutions are to be made after the complete configurations have been put together, as in (C). Difficulties arise if we do the substitution first. In each complete configuration the blanks would all have to be replaced by “ D ”, so that the complete configuration would not be expressed as a finite sequence of symbols.

If in the description of the machine II of § 3 we replace “ \mathfrak{o} ” by “ DAA ”, “ \mathfrak{e} ” by “ $DCCC$ ”, “ \mathfrak{q} ” by “ $DAAA$ ”, then the sequence (C) becomes:

$$DA : DCCC DCCC DA ADC DDC : DCCC DCCC DAA ADC DDC : \dots \quad (C_1)$$

(This is the sequence of symbols on F -squares.)

It is not difficult to see that if \mathcal{M} can be constructed, then so can \mathcal{M}' . The manner of operation of \mathcal{M}' could be made to depend on having the rules of operation (*i.e.*, the S.D) of \mathcal{M} written somewhere within itself (*i.e.* within \mathcal{M}'); each step could be carried out by referring to these rules. We have only to regard the rules as being capable of being taken out and exchanged for others and we have something very akin to the universal machine.

One thing is lacking: at present the machine \mathcal{M}' prints no figures. We may correct this by printing between each successive pair of complete configurations the figures which appear in the new configuration but not in the old. Then (C_1) becomes

$$DDA : 0 : 0 : DCCC DCCC DA ADC DDC : DCCC \dots \quad (C_2)$$

It is not altogether obvious that the E -squares leave enough room for the necessary “rough work”, but this is, in fact, the case.

The sequences of letters between the colons in expressions such as (C_1) may be used as standard descriptions of the complete configurations. When the letters are replaced by figures, as in § 5, we shall have a numerical

description of the complete configuration, which may be called its description number.

7. *Detailed description of the universal machine.*

A table is given below of the behaviour of this universal machine. The m -configurations of which the machine is capable are all those occurring in the first and last columns of the table, together with all those which occur when we write out the unabbreviated tables of those which appear in the table in the form of m -functions. *E.g.*, $e(\text{anf})$ appears in the table and is an m -function. Its unabbreviated table is (see p. 239)

$e(\text{anf})$	$\begin{cases} \text{o} \\ \text{not o} \end{cases}$	R	$e_1(\text{anf})$
		L	$e(\text{anf})$
$e_1(\text{anf})$	$\begin{cases} \text{Any} \\ \text{None} \end{cases}$	R, E, R	$e_1(\text{anf})$
			anf

Consequently $e_1(\text{anf})$ is an m -configuration of \mathcal{U} .

When \mathcal{U} is ready to start work the tape running through it bears on it the symbol o on an F -square and again o on the next E -square; after this, on F -squares only, comes the S.D of the machine followed by a double colon “::” (a single symbol, on an F -square). The S.D consists of a number of instructions, separated by semi-colons.

Each instruction consists of five consecutive parts

(i) “ D ” followed by a sequence of letters “ A ”. This describes the relevant m -configuration.

(ii) “ D ” followed by a sequence of letters “ C ”. This describes the scanned symbol.

(iii) “ D ” followed by another sequence of letters “ C ”. This describes the symbol into which the scanned symbol is to be changed.

(iv) “ L ”, “ R ”, or “ N ”, describing whether the machine is to move to left, right, or not at all.

(v) “ D ” followed by a sequence of letters “ A ”. This describes the final m -configuration.

The machine \mathcal{U} is to be capable of printing “ A ”, “ C ”, “ D ”, “ 0 ”, “ 1 ”, “ u ”, “ v ”, “ w ”, “ x ”, “ y ”, “ z ”. The S.D is formed from “;”, “ A ”, “ C ”, “ D ”, “ L ”, “ R ”, “ N ”.

Subsidiary skeleton table.

$\text{on}(\mathfrak{C}, a)$	$\left\{ \begin{array}{lll} \text{Not } A & R, R & \text{con}(\mathfrak{C}, a) \\ A & L, Pa, R & \text{con}_1(\mathfrak{C}, a) \end{array} \right.$
$\text{on}_1(\mathfrak{C}, a)$	$\left\{ \begin{array}{lll} A & R, Pa, R & \text{con}_1(\mathfrak{C}, a) \\ D & R, Pa, R & \text{con}_2(\mathfrak{C}, a) \end{array} \right.$
$\text{on}_2(\mathfrak{C}, a)$	$\left\{ \begin{array}{lll} C & R, Pa, R & \text{con}_2(\mathfrak{C}, a) \\ \text{Not } C & R, R & \mathfrak{C} \end{array} \right.$

$\text{con}(\mathfrak{C}, a)$. Starting from an F -square, S say, the sequence C of symbols describing a configuration closest on the right of S is marked out with letters a . $\rightarrow \mathfrak{C}$.

$\text{con}(\mathfrak{C},)$. In the final configuration the machine is scanning the square which is four squares to the right of the last square of C . C is left unmarked.

The table for \mathfrak{U} .

\mathfrak{b}	$\mathfrak{f}(\mathfrak{b}_1, \mathfrak{b}_1, ::)$
\mathfrak{b}_1	$R, R, P; : R, R, PD, R, R, PA \rightarrow \text{anf}$
anf	$\mathfrak{f}(\text{anf}_1, :)$
anf_1	$\text{con}(\mathfrak{k}om, y)$
$\mathfrak{k}om$	$\left\{ \begin{array}{lll} ; & R, Pz, L & \text{con}(\mathfrak{k}mp, x) \\ z & L, L & \mathfrak{k}om \\ \text{not } z \text{ nor } ; & L & \mathfrak{k}om \end{array} \right.$
$\mathfrak{k}mp$	$\text{cpc}(\mathfrak{e}(\mathfrak{k}om, x, y), \mathfrak{s}im, x, y)$

b. The machine prints $:DA$ on the F -squares after $:: \rightarrow \text{anf}$.

anf . The machine marks the configuration in the last complete configuration with y . $\rightarrow \mathfrak{k}om$.

$\mathfrak{k}om$. The machine finds the last semi-colon not marked with z . It marks this semi-colon with z and the configuration following it with x .

$\mathfrak{k}mp$. The machine compares the sequences marked x and y . It erases all letters x and y . $\rightarrow \mathfrak{s}im$ if they are alike. Otherwise $\rightarrow \mathfrak{k}om$.

anf . Taking the long view, the last instruction relevant to the last configuration is found. It can be recognised afterwards as the instruction following the last semi-colon marked z . $\rightarrow \mathfrak{s}im$.

sim	$f'(\text{sim}_1, \text{sim}_1, z)$	sim . The machine marks out the instructions. That part of the instructions which refers to operations to be carried out is marked with u , and the final m -configuration with y . The letters z are erased.
sim_1	$\text{con}(\text{sim}_2, \text{ })$	
sim_2	$\begin{cases} A & \text{sim}_3 \\ \text{not } A & R, Pu, R, R, R \end{cases}$	sim_3
sim_3	$\begin{cases} \text{not } A & L, Py \\ A & L, Py, R, R, R \end{cases}$	$\text{e}(mf^k, z)$
mf^k	$g(mf^k, \text{ })$	mf^k . The last complete configuration is marked out into four sections. The configuration is left unmarked. The symbol directly preceding it is marked with x . The remainder of the complete configuration is divided into two parts, of which the first is marked with v and the last with w . A colon is printed after the whole. $\rightarrow \text{sh}$.
mf_1^k	$\begin{cases} \text{not } A & R, R \\ A & L, L, L, L \end{cases}$	mf_1^k
mf_2^k	$\begin{cases} C & R, Px, L, L, L \\ : & \\ D & R, Px, L, L, L \end{cases}$	mf_2^k
mf_3^k	$\begin{cases} \text{not } : & R, Pv, L, L, L \\ : & \end{cases}$	mf_3^k
mf_4^k	$\text{con}\left(l(l(mf_5^k)), \text{ }\right)$	
mf_5^k	$\begin{cases} \text{Any} & R, Pw, R \\ \text{None} & P: \end{cases}$	mf_5^k
sh	$f(\text{sh}_1, \text{inst}, u)$	sh . The instructions (marked u) are examined. If it is found that they involve "Print 0" or "Print 1", then 0: or 1: is printed at the end.
sh_1	L, L, L	sh_2
sh_2	$\begin{cases} D & R, R, R, R \\ \text{not } D & \end{cases}$	sh_2
sh_3	$\begin{cases} C & R, R \\ \text{not } C & \end{cases}$	sh_4
sh_4	$\begin{cases} C & R, R \\ \text{not } C & \end{cases}$	sh_5
sh_5	$\begin{cases} C & \text{inst} \\ \text{not } C & \text{pe}_2(\text{inst}, 0, \text{ }) \end{cases}$	$\text{pe}_2(\text{inst}, 1, \text{ })$

inst		$g(l(inst_1), u)$	inst.	The next complete configuration is written down, carrying out the marked instructions. The letters u, v, w, x, y are erased. $\rightarrow anf$.
inst ₁	α	R, E	inst ₁ (α)	
inst ₁ (L)		ce ₅ (vv, v, y, x, u, w)		
inst ₁ (R)		ce ₅ (vv, v, x, u, y, w)		
inst ₁ (N)		ec ₅ (vv, v, x, y, u, w)		
vv		e(anf)		

8. Application of the diagonal process.

It may be thought that arguments which prove that the real numbers are not enumerable would also prove that the computable numbers and sequences cannot be enumerable*. It might, for instance, be thought that the limit of a sequence of computable numbers must be computable. This is clearly only true if the sequence of computable numbers is defined by some rule.

Or we might apply the diagonal process. "If the computable sequences are enumerable, let a_n be the n -th computable sequence, and let $\phi_n(m)$ be the m -th figure in a_n . Let β be the sequence with $1 - \phi_n(n)$ as its n -th figure. Since β is computable, there exists a number K such that $1 - \phi_n(n) = \phi_K(n)$ all n . Putting $n = K$, we have $1 = 2\phi_K(K)$, i.e. 1 is even. This is impossible. The computable sequences are therefore not enumerable".

The fallacy in this argument lies in the assumption that β is computable. It would be true if we could enumerate the computable sequences by finite means, but the problem of enumerating computable sequences is equivalent to the problem of finding out whether a given number is the D.N. of a circle-free machine, and we have no general process for doing this in a finite number of steps. In fact, by applying the diagonal process argument correctly, we can show that there cannot be any such general process.

The simplest and most direct proof of this is by showing that, if this general process exists, then there is a machine which computes β . This proof, although perfectly sound, has the disadvantage that it may leave the reader with a feeling that "there must be something wrong". The proof which I shall give has not this disadvantage, and gives a certain insight into the significance of the idea "circle-free". It depends not on constructing β , but on constructing β' , whose n -th figure is $\phi_n(n)$.

* Cf. Hobson, *Theory of functions of a real variable* (2nd ed., 1921), 87, 88.

Let us suppose that there is such a process; that is to say, that we can invent a machine \mathbb{Q} which, when supplied with the S.D. of any computing machine \mathcal{M} will test this S.D. and if \mathcal{M} is circular will mark the S.D. with the symbol “ u ” and if it is circle-free will mark it with “ s ”. By combining the machines \mathbb{Q} and \mathcal{U} we could construct a machine \mathbb{H} to compute the sequence β' . The machine \mathbb{Q} may require a tape. We may suppose that it uses the E -squares beyond all symbols on F -squares, and that when it has reached its verdict all the rough work done by \mathbb{Q} is erased.

The machine \mathbb{H} has its motion divided into sections. In the first $N-1$ sections, among other things, the integers $1, 2, \dots, N-1$ have been written down and tested by the machine \mathbb{Q} . A certain number, say $R(N-1)$, of them have been found to be the D.N.'s of circle-free machines. In the N -th section the machine \mathbb{Q} tests the number N . If N is satisfactory, *i.e.*, if it is the D.N. of a circle-free machine, then $R(N) = 1 + R(N-1)$ and the first $R(N)$ figures of the sequence of which a D.N. is N are calculated. The $R(N)$ -th figure of this sequence is written down as one of the figures of the sequence β' computed by \mathbb{H} . If N is not satisfactory, then $R(N) = R(N-1)$ and the machine goes on to the $(N+1)$ -th section of its motion.

From the construction of \mathbb{H} we can see that \mathbb{H} is circle-free. Each section of the motion of \mathbb{H} comes to an end after a finite number of steps. For, by our assumption about \mathbb{Q} , the decision as to whether N is satisfactory is reached in a finite number of steps. If N is not satisfactory, then the N -th section is finished. If N is satisfactory, this means that the machine $\mathcal{M}(N)$ whose D.N. is N is circle-free, and therefore its $R(N)$ -th figure can be calculated in a finite number of steps. When this figure has been calculated and written down as the $R(N)$ -th figure of β' , the N -th section is finished. Hence \mathbb{H} is circle-free.

Now let K be the D.N. of \mathbb{H} . What does \mathbb{H} do in the K -th section of its motion? It must test whether K is satisfactory, giving a verdict “ s ” or “ u ”. Since K is the D.N. of \mathbb{H} and since \mathbb{H} is circle-free, the verdict cannot be “ u ”. On the other hand the verdict cannot be “ s ”. For if it were, then in the K -th section of its motion \mathbb{H} would be bound to compute the first $R(K-1)+1 = R(K)$ figures of the sequence computed by the machine with K as its D.N. and to write down the $R(K)$ -th as a figure of the sequence computed by \mathbb{H} . The computation of the first $R(K)-1$ figures would be carried out all right, but the instructions for calculating the $R(K)$ -th would amount to “calculate the first $R(K)$ figures computed by H and write down the $R(K)$ -th”. This $R(K)$ -th figure would never be found. *I.e.*, \mathbb{H} is circular, contrary both to what we have found in the last paragraph and to the verdict “ s ”. Thus both verdicts are impossible and we conclude that there can be no machine \mathbb{Q} .

We can show further that *there can be no machine* \mathcal{E} *which, when supplied with the S.D of an arbitrary machine* \mathcal{M} , *will determine whether* \mathcal{M} *ever prints a given symbol* (0 say).

We will first show that, if there is a machine \mathcal{E} , then there is a general process for determining whether a given machine \mathcal{M} prints 0 infinitely often. Let \mathcal{M}_1 be a machine which prints the same sequence as \mathcal{M} , except that in the position where the first 0 printed by \mathcal{M} stands, \mathcal{M}_1 prints $\bar{0}$. \mathcal{M}_2 is to have the first two symbols 0 replaced by $\bar{0}$, and so on. Thus, if \mathcal{M} were to print

$$A B A 0 1 A A B 0 0 1 0 A B \dots,$$

then \mathcal{M}_1 would print

$$A B A \bar{0} 1 A A B 0 0 1 0 A B \dots$$

and \mathcal{M}_2 would print

$$A B A \bar{0} 1 A A B \bar{0} 0 1 0 A B \dots.$$

Now let \mathcal{F} be a machine which, when supplied with the S.D of \mathcal{M} , will write down successively the S.D of \mathcal{M} , of \mathcal{M}_1 , of \mathcal{M}_2 , ... (there is such a machine). We combine \mathcal{F} with \mathcal{E} and obtain a new machine, \mathcal{G} . In the motion of \mathcal{G} first \mathcal{F} is used to write down the S.D of \mathcal{M} , and then \mathcal{E} tests it, : 0 : is written if it is found that \mathcal{M} never prints 0; then \mathcal{F} writes the S.D of \mathcal{M}_1 , and this is tested, : 0 : being printed if and only if \mathcal{M}_1 never prints 0, and so on. Now let us test \mathcal{G} with \mathcal{E} . If it is found that \mathcal{G} never prints 0, then \mathcal{M} prints 0 infinitely often; if \mathcal{G} prints 0 sometimes, then \mathcal{M} does not print 0 infinitely often.

Similarly there is a general process for determining whether \mathcal{M} prints 1 infinitely often. By a combination of these processes we have a process for determining whether \mathcal{M} prints an infinity of figures, *i.e.* we have a process for determining whether \mathcal{M} is circle-free. There can therefore be no machine \mathcal{E} .

The expression "there is a general process for determining ..." has been used throughout this section as equivalent to "there is a machine which will determine ...". This usage can be justified if and only if we can justify our definition of "computable". For each of these "general process" problems can be expressed as a problem concerning a general process for determining whether a given integer n has a property $G(n)$ [*e.g.* $G(n)$ might mean " n is satisfactory" or " n is the Gödel representation of a provable formula"], and this is equivalent to computing a number whose n -th figure is 1 if $G(n)$ is true and 0 if it is false.

9. *The extent of the computable numbers.*

No attempt has yet been made to show that the “computable” numbers include all numbers which would naturally be regarded as computable. All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. The real question at issue is “What are the possible processes which can be carried out in computing a number?”

The arguments which I shall use are of three kinds.

(a) A direct appeal to intuition.

(b) A proof of the equivalence of two definitions (in case the new definition has a greater intuitive appeal).

(c) Giving examples of large classes of numbers which are computable.

Once it is granted that computable numbers are all “computable”, several other propositions of the same character follow. In particular, it follows that, if there is a general process for determining whether a formula of the Hilbert function calculus is provable, then the determination can be carried out by a machine.

I. [Type (a)]. This argument is only an elaboration of the ideas of § 1.

Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child’s arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. 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†. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of symbols in the place of single symbols. Thus an Arabic numeral such as

† If we regard a symbol as literally printed on a square we may suppose that the square is $0 < x < 1$, $0 < y < 1$. The symbol is defined as a set of points in this square, *viz.* the set occupied by printer’s ink. If these sets are restricted to be measurable, we can define the “distance” between two symbols as the cost of transforming one symbol into the other if the cost of moving unit area of printer’s ink unit distance is unity, and there is an infinite supply of ink at $x = 2$, $y = 0$. With this topology the symbols form a conditionally compact space.

17 or 9999999999999999 is normally treated as a single symbol. Similarly in any European language words are treated as single symbols (Chinese, however, attempts to have an enumerable infinity of symbols). The differences from our point of view between the single and compound symbols is that the compound symbols, if they are too lengthy, cannot be observed at one glance. This is in accordance with experience. We cannot tell at a glance whether 9999999999999999 and 9999999999999999 are the same.

The behaviour of the computer at any moment is determined by the symbols which he is observing, 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 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. Again, the restriction is not one which seriously affects computation, since the use of more complicated states of mind can be avoided by writing more symbols on the tape.

Let us imagine the operations performed by the computer to be split up into "simple operations" which are so elementary that it is not easy to imagine them further divided. Every such operation consists of some change of the physical system consisting of the computer and his tape. We know the state of the system if we know the sequence of symbols on the tape, which of these are observed by the computer (possibly with a special order), and the state of mind of the computer. We may suppose that in a simple operation not more than one symbol is altered. Any other changes can be split up into simple changes of this kind. The situation in regard to the squares whose symbols may be altered in this way is the same as in regard to the observed squares. We may, therefore, without loss of generality, assume that the squares whose symbols are changed are always "observed" squares.

Besides these changes of symbols, the simple operations must include changes of distribution of observed squares. The new observed squares must be immediately recognisable by the computer. I think it is reasonable to suppose that they can only be squares whose distance from the closest of the immediately previously observed squares does not exceed a certain fixed amount. Let us say that each of the new observed squares is within L squares of an immediately previously observed square.

In connection with "immediate recognisability", it may be thought that there are other kinds of square which are immediately recognisable. In particular, squares marked by special symbols might be taken as imme-

dately recognisable. Now if these squares are marked only by single symbols there can be only a finite number of them, and we should not upset our theory by adjoining these marked squares to the observed squares. If, on the other hand, they are marked by a sequence of symbols, we cannot regard the process of recognition as a simple process. This is a fundamental point and should be illustrated. In most mathematical papers the equations and theorems are numbered. Normally the numbers do not go beyond (say) 1000. It is, therefore, possible to recognise a theorem at a glance by its number. But if the paper was very long, we might reach Theorem 157767733443477; then, further on in the paper, we might find "... hence (applying Theorem 157767733443477) we have ...". In order to make sure which was the relevant theorem we should have to compare the two numbers figure by figure, possibly ticking the figures off in pencil to make sure of their not being counted twice. If in spite of this it is still thought that there are other "immediately recognisable" squares, it does not upset my contention so long as these squares can be found by some process of which my type of machine is capable. This idea is developed in III below.

The simple operations must therefore include:

- (a) Changes of the symbol on one of the observed squares.
- (b) Changes of one of the squares observed to another square within L squares of one of the previously observed squares.

It may be that some of these changes necessarily involve a change of state of mind. The most general single operation must therefore be taken to be one of the following:

- (A) A possible change (a) of symbol together with a possible change of state of mind.
- (B) A possible change (b) of observed squares, together with a possible change of state of mind.

The operation actually performed is determined, as has been suggested on p. 250, by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out.

We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an " m -configuration" of the machine. The machine scans B squares corresponding to the B squares observed by the computer. In any move the machine can change a symbol on a scanned square or can change any one of the scanned squares to another square distant not more than L squares from one of the other scanned

squares. The move which is done, and the succeeding configuration, are determined by the scanned symbol and the m -configuration. The machines just described do not differ very essentially from computing machines as defined in § 2, and corresponding to any machine of this type a computing machine can be constructed to compute the same sequence, that is to say the sequence computed by the computer.

II. [Type (b)].

If the notation of the Hilbert functional calculus[†] is modified so as to be systematic, and so as to involve only a finite number of symbols, it becomes possible to construct an automatic[‡] machine \mathcal{H} , which will find all the provable formulae of the calculus[§].

Now let a be a sequence, and let us denote by $G_a(x)$ the proposition “The x -th figure of a is 1”, so that^{||} $\neg G_a(x)$ means “The x -th figure of a is 0”. Suppose further that we can find a set of properties which define the sequence a and which can be expressed in terms of $G_a(x)$ and of the propositional functions $N(x)$ meaning “ x is a non-negative integer” and $F(x, y)$ meaning “ $y = x + 1$ ”. When we join all these formulae together conjunctively, we shall have a formula, \mathfrak{U} say, which defines a . The terms of \mathfrak{U} must include the necessary parts of the Peano axioms, viz.,

$$(\exists u) N(u) \& (x) (N(x) \rightarrow (\exists y) F(x, y)) \& (F(x, y) \rightarrow N(y)),$$

which we will abbreviate to P .

When we say “ \mathfrak{U} defines a ”, we mean that $\neg \mathfrak{U}$ is not a provable formula, and also that, for each n , one of the following formulae (A_n) or (B_n) is provable.

$$\mathfrak{U} \& F^{(n)} \rightarrow G_a(u^{(n)}), \quad (A_n) \mathbb{T}$$

$$\mathfrak{U} \& F^{(n)} \rightarrow (\neg G_a(u^{(n)})), \quad (B_n),$$

where $F^{(n)}$ stands for $F(u, u') \& F(u', u'') \& \dots F(u^{(n-1)}, u^{(n)})$.

[†] The expression “the functional calculus” is used throughout to mean the *restricted* Hilbert functional calculus.

[‡] It is most natural to construct first a choice machine (§ 2) to do this. But it is then easy to construct the required automatic machine. We can suppose that the choices are always choices between two possibilities 0 and 1. Each proof will then be determined by a sequence of choices i_1, i_2, \dots, i_n ($i_1 = 0$ or 1, $i_2 = 0$ or 1, ..., $i_n = 0$ or 1), and hence the number $2^n + i_1 2^{n-1} + i_2 2^{n-2} + \dots + i_n$ completely determines the proof. The automatic machine carries out successively proof 1, proof 2, proof 3,

[§] The author has found a description of such a machine.

^{||} The negation sign is written before an expression and not over it.

[¶] A sequence of r primes is denoted by $^{(r)}$.

I say that α is then a computable sequence: a machine \mathcal{K}_α to compute α can be obtained by a fairly simple modification of \mathcal{K} .

We divide the motion of \mathcal{K}_α into sections. The n -th section is devoted to finding the n -th figure of α . After the $(n-1)$ -th section is finished a double colon $::$ is printed after all the symbols, and the succeeding work is done wholly on the squares to the right of this double colon. The first step is to write the letter “ A ” followed by the formula (A_n) and then “ B ” followed by (B_n) . The machine \mathcal{K}_α then starts to do the work of \mathcal{K} , but whenever a provable formula is found, this formula is compared with (A_n) and with (B_n) . If it is the same formula as (A_n) , then the figure “1” is printed, and the n -th section is finished. If it is (B_n) , then “0” is printed and the section is finished. If it is different from both, then the work of \mathcal{K} is continued from the point at which it had been abandoned. Sooner or later one of the formulae (A_n) or (B_n) is reached; this follows from our hypotheses about α and \mathfrak{U} , and the known nature of \mathcal{K} . Hence the n -th section will eventually be finished. \mathcal{K}_α is circle-free; α is computable.

It can also be shown that the numbers α definable in this way by the use of axioms include all the computable numbers. This is done by describing computing machines in terms of the function calculus.

It must be remembered that we have attached rather a special meaning to the phrase “ \mathfrak{U} defines α ”. The computable numbers do not include all (in the ordinary sense) definable numbers. Let δ be a sequence whose n -th figure is 1 or 0 according as n is or is not satisfactory. It is an immediate consequence of the theorem of § 8 that δ is not computable. It is (so-far as we know at present) possible that any assigned number of figures of δ can be calculated, but not by a uniform process. When sufficiently many figures of δ have been calculated, an essentially new method is necessary in order to obtain more figures.

III. This may be regarded as a modification of I or as a corollary of II.

We suppose, as in I, that the computation is carried out on a tape; but we avoid introducing the “state of mind” by considering a more physical and definite counterpart of it. It is always possible for the computer 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. Thus the state of progress of the computation at any stage is completely determined by the note of

instructions and the symbols on the tape. That is, the state of the system may be described by a single expression (sequence of symbols), consisting of the symbols on the tape followed by Δ (which we suppose not to appear elsewhere) and then by the note of instructions. This expression may be called the "state formula". We know that the state formula at any given stage is determined by the state formula before the last step was made, and we assume that the relation of these two formulae is expressible in the functional calculus. In other words, we assume that there is an axiom \mathfrak{A} which expresses the rules governing the behaviour of the computer, in terms of the relation of the state formula at any stage to the state formula at the preceding stage. If this is so, we can construct a machine to write down the successive state formulae, and hence to compute the required number.

10. Examples of large classes of numbers which are computable.

It will be useful to begin with definitions of a computable function of an integral variable and of a computable variable, etc. There are many equivalent ways of defining a computable function of an integral variable. The simplest is, possibly, as follows. If γ is a computable sequence in which 0 appears infinitely† often, and n is an integer, then let us define $\xi(\gamma, n)$ to be the number of figures 1 between the n -th and the $(n+1)$ -th figure 0 in γ . Then $\phi(n)$ is computable if, for all n and some γ , $\phi(n) = \xi(\gamma, n)$. An equivalent definition is this. Let $H(x, y)$ mean $\phi(x) = y$. Then, if we can find a contradiction-free axiom \mathfrak{A}_ϕ , such that $\mathfrak{A}_\phi \rightarrow P$, and if for each integer n there exists an integer N , such that

$$\mathfrak{A}_\phi \& F^{(N)} \rightarrow H(u^{(n)}, u^{(\phi(n))}),$$

and such that, if $m \neq \phi(n)$, then, for some N' ,

$$\mathfrak{A}_\phi \& F^{(N')} \rightarrow (-H(u^{(n)}, u^{(m)}),$$

then ϕ may be said to be a computable function.

We cannot define general computable functions of a real variable, since there is no general method of describing a real number, but we can define a computable function of a computable variable. If n is satisfactory, let γ_n be the number computed by $\mathcal{M}(n)$, and let

$$a_n = \tan \left(\pi(\gamma_n - \frac{1}{2}) \right),$$

† If \mathcal{M} computes γ , then the problem whether \mathcal{M} prints 0 infinitely often is of the same character as the problem whether \mathcal{M} is circle-free.

unless $\gamma_n = 0$ or $\gamma_n = 1$, in either of which cases $a_n = 0$. Then, as n runs through the satisfactory numbers, a_n runs through the computable numbers†. Now let $\phi(n)$ be a computable function which can be shown to be such that for any satisfactory argument its value is satisfactory‡. Then the function f , defined by $f(a_n) = a_{\phi(n)}$, is a computable function and all computable functions of a computable variable are expressible in this form.

Similar definitions may be given of computable functions of several variables, computable-valued functions of an integral variable, etc.

I shall enunciate a number of theorems about computability, but I shall prove only (ii) and a theorem similar to (iii).

(i) A computable function of a computable function of an integral or computable variable is computable.

(ii) Any function of an integral variable defined recursively in terms of computable functions is computable. *I.e.* if $\phi(m, n)$ is computable, and r is some integer, then $\eta(n)$ is computable, where

$$\begin{aligned}\eta(0) &= r, \\ \eta(n) &= \phi(n, \eta(n-1)).\end{aligned}$$

(iii) If $\phi(m, n)$ is a computable function of two integral variables, then $\phi(n, n)$ is a computable function of n .

(iv) If $\phi(n)$ is a computable function whose value is always 0 or 1, then the sequence whose n -th figure is $\phi(n)$ is computable.

Dedekind's theorem does not hold in the ordinary form if we replace "real" throughout by "computable". But it holds in the following form:

(v) If $G(a)$ is a propositional function of the computable numbers and

$$(a) \quad (\exists a)(\exists \beta) \{ G(a) \& \{ -G(\beta) \} \},$$

$$(b) \quad G(a) \& \{ -G(\beta) \} \rightarrow (a < \beta),$$

and there is a general process for determining the truth value of $G(a)$, then

† A function a_n may be defined in many other ways so as to run through the computable numbers.

‡ Although it is not possible to find a general process for determining whether a given number is satisfactory, it is often possible to show that certain classes of numbers are satisfactory.

there is a computable number ξ such that

$$\begin{aligned} G(a) \rightarrow a &\leq \xi, \\ -G(a) \rightarrow a &\geq \xi. \end{aligned}$$

In other words, the theorem holds for any section of the computables such that there is a general process for determining to which class a given number belongs.

Owing to this restriction of Dedekind's theorem, we cannot say that a computable bounded increasing sequence of computable numbers has a computable limit. This may possibly be understood by considering a sequence such as

$$-1, -\frac{1}{2}, -\frac{1}{4}, -\frac{1}{8}, -\frac{1}{16}, \frac{1}{2}, \dots$$

On the other hand, (v) enables us to prove

(vi) If α and β are computable and $\alpha < \beta$ and $\phi(\alpha) < 0 < \phi(\beta)$, where $\phi(a)$ is a computable increasing continuous function, then there is a unique computable number γ , satisfying $\alpha < \gamma < \beta$ and $\phi(\gamma) = 0$.

Computable convergence.

We shall say that a sequence β_n of computable numbers *converges computably* if there is a computable integral valued function $N(\epsilon)$ of the computable variable ϵ , such that we can show that, if $\epsilon > 0$ and $n > N(\epsilon)$ and $m > N(\epsilon)$, then $|\beta_n - \beta_m| < \epsilon$.

We can then show that

(vii) A power series whose coefficients form a computable sequence of computable numbers is computably convergent at all computable points in the interior of its interval of convergence.

(viii) The limit of a computably convergent sequence is computable.

And with the obvious definition of "uniformly computably convergent":

(ix) The limit of a uniformly computably convergent computable sequence of computable functions is a computable function. Hence

(x) The sum of a power series whose coefficients form a computable sequence is a computable function in the interior of its interval of convergence.

From (viii) and $\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \dots)$ we deduce that π is computable.

From $e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \dots$ we deduce that e is computable.

From (vi) we deduce that all real algebraic numbers are computable.

From (vi) and (x) we deduce that the real zeros of the Bessel functions are computable.

Proof of (ii).

Let $H(x, y)$ mean " $\eta(x) = y$ ", and let $K(x, y, z)$ mean " $\phi(x, y) = z$ ". \mathfrak{A}_ϕ is the axiom for $\phi(x, y)$. We take \mathfrak{A}_η to be

$$\begin{aligned}
& \mathfrak{A}_\phi \ \& \ P \ \& \ \left(F(x, y) \rightarrow G(x, y) \right) \ \& \ \left(G(x, y) \ \& \ G(y, z) \rightarrow G(x, z) \right) \\
& \ \& \ \left(F^{(r)} \rightarrow H(u, u^{(r)}) \right) \ \& \ \left(F(v, w) \ \& \ H(v, x) \ \& \ K(w, x, z) \rightarrow H(w, z) \right) \\
& \ \& \ \left[H(w, z) \ \& \ G(z, t) \vee G(t, z) \rightarrow \left(-H(w, t) \right) \right].
\end{aligned}$$

I shall not give the proof of consistency of \mathfrak{A}_n . Such a proof may be constructed by the methods used in Hilbert and Bernays, *Grundlagen der Mathematik* (Berlin, 1934), p. 209 *et seq.* The consistency is also clear from the meaning.

Suppose that, for some n, N , we have shown

$$\mathfrak{A}_\eta \& F^{(N)} \rightarrow H(u^{(n-1)}, u^{(\eta(n-1))}),$$

then, for some M ,

$$\mathfrak{A}_\phi \& F^{(M)} \rightarrow K(u^{(n)}, u^{(\eta(n-1))}, u^{(\eta(n))}),$$

$$\mathfrak{U}_\eta \& F^{(M)} \rightarrow F(u^{(n-1)}, u^{(n)}) \& H(u^{(n-1)}, u^{(\eta(n-1))}) \\ \& \& K(u^{(n)}, u^{(\eta(n-1))}, u^{(\eta(n))}),$$

and

$$\mathfrak{A}_\eta \& F^{(M)} \rightarrow [F(u^{(n-1)}, u^{(n)}) \& H(u^{(n-1)}, u^{(\eta(n-1))}) \\ \& \& K(u^{(n)}, u^{(\eta(n-1))}, u^{(\eta(n))}) \rightarrow H(u^{(n)}, u^{(\eta(n))})].$$

Hence

$$\mathfrak{A}_\eta \& F^{(M)} \rightarrow H(u^{(n)}, u^{(\eta(n))}).$$

Also

$$\mathfrak{U}_n \& F^{(r)} \rightarrow H(u, u^{(\eta(0))}).$$

Hence for each n some formula of the form

$$\mathfrak{U}_n \& F^{(M)} \rightarrow H(u^{(n)}, u^{(\eta(n))})$$

is provable. Also, if $M' \geq M$ and $M' \geq m$ and $m \neq \eta(u)$, then

$$\mathfrak{U}_n \& F^{(M')} \rightarrow G(u^{\eta((n))}, u^{(m)}) \vee G(u^{(m)}, u^{(\eta(n))})$$

and

$$\begin{aligned} \mathfrak{U}_\eta \& F^{(M)} \rightarrow & \left[\{G(u^{(\eta(n))}, u^{(m)}) \nu G(u^{(m)}, u^{(\eta(n))}) \right. \\ & \left. \& H(u^{(n)}, u^{(\eta(n))}) \} \rightarrow (-H(u^{(n)}, u^{(m)})) \right]. \end{aligned}$$

Hence $\mathfrak{U}_\eta \& F^{(M)} \rightarrow (-H(u^{(n)}, u^{(m)})).$

The conditions of our second definition of a computable function are therefore satisfied. Consequently η is a computable function.

Proof of a modified form of (iii).

Suppose that we are given a machine \mathfrak{N} , which, starting with a tape bearing on it $\epsilon \epsilon$ followed by a sequence of any number of letters “ F ” on F -squares and in the m -configuration b , will compute a sequence γ_n depending on the number n of letters “ F ”. If $\phi_n(m)$ is the m -th figure of γ_n , then the sequence β whose n -th figure is $\phi_n(n)$ is computable.

We suppose that the table for \mathfrak{N} has been written out in such a way that in each line only one operation appears in the operations column. We also suppose that Ξ , Θ , $\bar{0}$, and $\bar{1}$ do not occur in the table, and we replace ϵ throughout by Θ , 0 by $\bar{0}$, and 1 by $\bar{1}$. Further substitutions are then made. Any line of form

$$\mathfrak{U} \quad a \quad P\bar{0} \quad \mathfrak{B}$$

we replace by

$$\mathfrak{U} \quad a \quad P\bar{0} \quad \text{re}(\mathfrak{B}, u, h, k)$$

and any line of the form

$$\mathfrak{U} \quad a \quad P\bar{1} \quad \mathfrak{B}$$

by $\mathfrak{U} \quad a \quad P\bar{1} \quad \text{re}(\mathfrak{B}, v, h, k)$

and we add to the table the following lines:

$$\begin{array}{llll} u & & & \text{pe}(u_1, 0) \\ u_1 & R, Pk, R, P\Theta, R, P\Theta & & u_2 \\ u_2 & & & \text{re}(u_3, u_3, k, h) \\ u_3 & & & \text{pe}(u_2, F) \end{array}$$

and similar lines with v for u and 1 for 0 together with the following line

$$c \quad R, P\Xi, R, Ph \quad b.$$

We then have the table for the machine \mathfrak{N}' which computes β . The initial m -configuration is c , and the initial scanned symbol is the second ϵ .

11. *Application to the Entscheidungsproblem.*

The results of §8 have some important applications. In particular, they can be used to show that the Hilbert Entscheidungsproblem can have no solution. For the present I shall confine myself to proving this particular theorem. For the formulation of this problem I must refer the reader to Hilbert and Ackermann's *Grundzüge der Theoretischen Logik* (Berlin, 1931), chapter 3.

I propose, therefore, to show that there can be no general process for determining whether a given formula \mathfrak{A} of the functional calculus \mathbf{K} is provable, *i.e.* that there can be no machine which, supplied with any one \mathfrak{A} of these formulae, will eventually say whether \mathfrak{A} is provable.

It should perhaps be remarked that what I shall prove is quite different from the well-known results of Gödel†. Gödel has shown that (in the formalism of Principia Mathematica) there are propositions \mathfrak{A} such that neither \mathfrak{A} nor $-\mathfrak{A}$ is provable. As a consequence of this, it is shown that no proof of consistency of Principia Mathematica (or of \mathbf{K}) can be given within that formalism. On the other hand, I shall show that there is no general method which tells whether a given formula \mathfrak{A} is provable in \mathbf{K} , or, what comes to the same, whether the system consisting of \mathbf{K} with $-\mathfrak{A}$ adjoined as an extra axiom is consistent.

If the negation of what Gödel has shown had been proved, *i.e.* if, for each \mathfrak{A} , either \mathfrak{A} or $-\mathfrak{A}$ is provable, then we should have an immediate solution of the Entscheidungsproblem. For we can invent a machine \mathcal{K} which will prove consecutively all provable formulae. Sooner or later \mathcal{K} will reach either \mathfrak{A} or $-\mathfrak{A}$. If it reaches \mathfrak{A} , then we know that \mathfrak{A} is provable. If it reaches $-\mathfrak{A}$, then, since \mathbf{K} is consistent (Hilbert and Ackermann, p. 65), we know that \mathfrak{A} is not provable.

Owing to the absence of integers in \mathbf{K} the proofs appear somewhat lengthy. The underlying ideas are quite straightforward.

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

The interpretations of the propositional functions involved are as follows :

$R_{S_t}(x, y)$ is to be interpreted as "in the complete configuration x (of \mathcal{M}) the symbol on the square y is S ".

† *Loc. cit.*

$I(x, y)$ is to be interpreted as “in the complete configuration x the square y is scanned”.

$K_{q_m}(x)$ is to be interpreted as “in the complete configuration x the m -configuration is q_m .

$F(x, y)$ is to be interpreted as “ y is the immediate successor of x ”.

$\text{Inst}\{q_i S_j S_k L q_l\}$ is to be an abbreviation for

$$(x, y, x', y') \left\{ \begin{array}{l} \left(R_{S_j}(x, y) \& I(x, y) \& K_{q_i}(x) \& F(x, x') \& F(y', y) \right) \\ \rightarrow \left(I(x', y') \& R_{S_k}(x', y) \& K_{q_i}(x') \right. \\ \left. \& (z) \left[F(y', z) \vee \left(R_{S_j}(x, z) \rightarrow R_{S_k}(x', z) \right) \right] \right) \end{array} \right\}.$$

$\text{Inst}\{q_i S_j S_k R q_l\}$ and $\text{Inst}\{q_i S_j S_k N q_l\}$

are to be abbreviations for other similarly constructed expressions.

Let us put the description of \mathcal{M} into the first standard form of § 6. This description consists of a number of expressions such as “ $q_i S_j S_k L q_l$ ” (or with R or N substituted for L). Let us form all the corresponding expressions such as $\text{Inst}\{q_i S_j S_k L q_l\}$ and take their logical sum. This we call $\text{Des}(\mathcal{M})$.

The formula $\text{Un}(\mathcal{M})$ is to be

$$\begin{aligned} (\exists u) \left[N(u) \& (x) \left(N(x) \rightarrow (\exists x') F(x, x') \right) \right. \\ & \& (y, z) \left(F(y, z) \rightarrow N(y) \& N(z) \right) \& (y) R_{S_0}(u, y) \\ & \& I(u, u) \& K_{q_1}(u) \& \text{Des}(\mathcal{M}) \left. \right] \\ & \rightarrow (\exists s) (\exists t) [N(s) \& N(t) \& R_{S_1}(s, t)]. \end{aligned}$$

$[N(u) \& \dots \& \text{Des}(\mathcal{M})]$ may be abbreviated to $A(\mathcal{M})$.

When we substitute the meanings suggested on p. 259–60 we find that $\text{Un}(\mathcal{M})$ has the interpretation “in some complete configuration of \mathcal{M} , S_1 (i.e. 0) appears on the tape”. Corresponding to this I prove that

(a) If S_1 appears on the tape in some complete configuration of \mathcal{M} , then $\text{Un}(\mathcal{M})$ is provable.

(b) If $\text{Un}(\mathcal{M})$ is provable, then S_1 appears on the tape in some complete configuration of \mathcal{M} .

When this has been done, the remainder of the theorem is trivial.

LEMMA 1. If S_1 appears on the tape in some complete configuration of M , then $Un(M)$ is provable.

We have to show how to prove $Un(\mathcal{M})$. Let us suppose that in the n -th complete configuration the sequence of symbols on the tape is $S_{r(n,0)}, S_{r(n,1)}, \dots, S_{r(n,n)}$, followed by nothing but blanks, and that the scanned symbol is the $i(n)$ -th, and that the m -configuration is $q_{k(n)}$. Then we may form the proposition

which we may abbreviate to CC_n .

As before, $F(u, u')$ & $F(u', u'')$ & ... & $F(u^{(r-1)}, u^{(r)})$ is abbreviated to $F^{(r)}$.

I shall show that all formulae of the form $A(\mathcal{M}) \& F^{(n)} \rightarrow CC_n$ (abbreviated to CF_n) are provable. The meaning of CF_n is “The n -th complete configuration of \mathcal{M} is so and so”, where “so and so” stands for the actual n -th complete configuration of \mathcal{M} . That CF_n should be provable is therefore to be expected.

CF_0 is certainly provable, for in the complete configuration the symbols are all blanks, the m -configuration is q_1 , and the scanned square is u , i.e. CC_0 is

$$(y) R_{S_0}(u, y) \& I(u, u) \& K_{q_1}(u).$$

$A(\mathcal{M}) \rightarrow CC_0$ is then trivial.

We next show that $CF_n \rightarrow CF_{n+1}$ is provable for each n . There are three cases to consider, according as in the move from the n -th to the $(n+1)$ -th configuration the machine moves to left or to right or remains stationary. We suppose that the first case applies, *i.e.* the machine moves to the left. A similar argument applies in the other cases. If $r(n, i(n)) = a$, $r(n+1, i(n+1)) = c$, $k(i(n)) = b$, and $k(i(n+1)) = d$, then $\text{Des}(\mathcal{M})$ must include $\text{Inst}\{q_a, S_b, S_d, Lq_c\}$ as one of its terms, *i.e.*

Des (\mathcal{M}) \rightarrow **Inst** $\{q_a, S_b, S_d, L, q_c\}$:

Hence $A(\mathcal{M}) \& F^{(n+1)} \rightarrow \text{Inst}\{q_a S_b S_d L q_c\} \& F^{(n+1)}$

But $\text{Inst}\{q_a S_b S_d L q_c\} \& F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1})$

is provable, and so therefore is

$$A(\mathcal{M}) \& F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1})$$

and $(A(\mathcal{M}) \& F^{(n)} \rightarrow CC_n) \rightarrow (A(\mathcal{M}) \& F^{(n+1)} \rightarrow CC_{n+1})$,

i.e. $CF_n \rightarrow CF_{n+1}$.

CF_n is provable for each n . Now it is the assumption of this lemma that S_1 appears somewhere, in some complete configuration, in the sequence of symbols printed by \mathcal{M} ; that is, for some integers N, K , CC_N has $R_{S_1}(u^{(N)}, u^{(K)})$ as one of its terms, and therefore $CC_N \rightarrow R_{S_1}(u^{(N)}, u^{(K)})$ is provable. We have then

$$CC_N \rightarrow R_{S_1}(u^{(N)}, u^{(K)})$$

and $A(\mathcal{M}) \& F^{(N)} \rightarrow CC^N$.

We also have

$$(\exists u) A(\mathcal{M}) \rightarrow (\exists u)(\exists u') \dots (\exists u^{(N')}) (A(\mathcal{M}) \& F^{(N)}),$$

where $N' = \max(N, K)$. And so

$$(\exists u) A(\mathcal{M}) \rightarrow (\exists u)(\exists u') \dots (\exists u^{(N')}) R_{S_1}(u^{(N)}, u^{(K)}),$$

$$(\exists u) A(\mathcal{M}) \rightarrow (\exists u^{(N)}) (\exists u^{(K)}) R_{S_1}(u^{(N)}, u^{(K)}),$$

$$(\exists u) A(\mathcal{M}) \rightarrow (\exists s)(\exists t) R_{S_1}(s, t),$$

i.e. $Un(\mathcal{M})$ is provable.

This completes the proof of Lemma 1.

LEMMA 2. *If $Un(\mathcal{M})$ is provable, then S_1 appears on the tape in some complete configuration of \mathcal{M} .*

If we substitute any propositional functions for function variables in a provable formula, we obtain a true proposition. In particular, if we substitute the meanings tabulated on pp. 259–260 in $Un(\mathcal{M})$, we obtain a true proposition with the meaning “ S_1 appears somewhere on the tape in some complete configuration of \mathcal{M} ”.

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 $Un(\mathcal{M})$ is provable. By Lemmas 1 and 2, this implies that there is a process for determining whether \mathcal{M} ever prints 0, and this is impossible, by §8. Hence the Entscheidungsproblem cannot be solved.

In view of the large number of particular cases of solutions of the Entscheidungsproblem for formulae with restricted systems of quantors, it
[50]

is interesting to express $Un(\mathcal{M})$ in a form in which all quantors are at the beginning. $Un(\mathcal{M})$ is, in fact, expressible in the form

$$(u)(\exists x)(w)(\exists u_1) \dots (\exists u_n) \mathfrak{B}, \quad (I)$$

where \mathfrak{B} contains no quantors, and $n = 6$. By unimportant modifications we can obtain a formula, with all essential properties of $Un(\mathcal{M})$, which is of form (I) with $n = 5$.

Added 28 August, 1936.

APPENDIX.

Computability and effective calculability

The theorem that all effectively calculable (λ -definable) sequences are computable and its converse are proved below in outline. It is assumed that the terms "well-formed formula" (W.F.F.) and "conversion" as used by Church and Kleene are understood. In the second of these proofs the existence of several formulae is assumed without proof; these formulae may be constructed straightforwardly with the help of, *e.g.*, the results of Kleene in "A theory of positive integers in formal logic", *American Journal of Math.*, 57 (1935), 153-173, 219-244.

The W.F.F. representing an integer n will be denoted by N_n . We shall say that a sequence γ whose n -th figure is $\phi_\gamma(n)$ is λ -definable or effectively calculable if $1 + \phi_\gamma(n)$ is a λ -definable function of n , *i.e.* if there is a W.F.F. M_γ such that, for all integers n ,

$$\{M_\gamma\}(N_n) \text{ conv } N_{\phi_\gamma(n)+1},$$

i.e. $\{M_\gamma\}(N_n)$ is convertible into $\lambda xy.x(x(y))$ or into $\lambda xy.x(y)$ according as the n -th figure of λ is 1 or 0.

To show that every λ -definable sequence γ is computable, we have to show how to construct a machine to compute γ . For use with machines it is convenient to make a trivial modification in the calculus of conversion. This alteration consists in using x, x', x'', \dots as variables instead of a, b, c, \dots . We now construct a machine \mathcal{L} which, when supplied with the formula M_γ , writes down the sequence γ . The construction of \mathcal{L} is somewhat similar to that of the machine \mathcal{K} which proves all provable formulae of the functional calculus. We first construct a choice machine \mathcal{L}_1 , which, if supplied with a W.F.F., M say, and suitably manipulated, obtains any formula into which M is convertible. \mathcal{L}_1 can then be modified so as to yield an automatic machine \mathcal{L}_2 which obtains successively all the formulae

into which M is convertible (cf. foot-note p. 252). The machine \mathcal{L} includes \mathcal{L}_2 as a part. The motion of the machine \mathcal{L} when supplied with the formula M_γ is divided into sections of which the n -th is devoted to finding the n -th figure of γ . The first stage in this n -th section is the formation of $\{M_\gamma\}(N_n)$. This formula is then supplied to the machine \mathcal{L}_2 , which converts it successively into various other formulae. Each formula into which it is convertible eventually appears, and each, as it is found, is compared with

$$\lambda x \left[\lambda x' \left[\{x\}(\{x\}(x')) \right] \right], \text{ i.e. } N_2,$$

and with

$$\lambda x \left[\lambda x' [\{x\}(x')] \right], \text{ i.e. } N_1.$$

If it is identical with the first of these, then the machine prints the figure 1 and the n -th section is finished. If it is identical with the second, then 0 is printed and the section is finished. If it is different from both, then the work of \mathcal{L}_2 is resumed. By hypothesis, $\{M_\gamma\}(N_n)$ is convertible into one of the formulae N_2 or N_1 ; consequently the n -th section will eventually be finished, i.e. the n -th figure of γ will eventually be written down.

To prove that every computable sequence γ is λ -definable, we must show how to find a formula M_γ such that, for all integers n ,

$$\{M_\gamma\}(N_n) \text{ conv } N_{1+\phi_{\gamma}(n)}.$$

Let \mathcal{M} be a machine which computes γ and let us take some description of the complete configurations of \mathcal{M} by means of numbers, e.g. we may take the D.N. of the complete configuration as described in § 6. Let $\xi(n)$ be the D.N. of the n -th complete configuration of \mathcal{M} . The table for the machine \mathcal{M} gives us a relation between $\xi(n+1)$ and $\xi(n)$ of the form

$$\xi(n+1) = \rho_\gamma(\xi(n)),$$

where ρ_γ is a function of very restricted, although not usually very simple, form: it is determined by the table for \mathcal{M} . ρ_γ is λ -definable (I omit the proof of this), i.e. there is a W.F.F. A_γ such that, for all integers n ,

$$\{A_\gamma\}(N_{\xi(n)}) \text{ conv } N_{\xi(n+1)}.$$

Let U stand for

$$\lambda u \left[\{u\}(A_\gamma) \right] (N_r),$$

where $r = \xi(0)$; then, for all integers n ,

$$\{U_\gamma\}(N_n) \text{ conv } N_{\xi(n)}.$$

It may be proved that there is a formula V such that

$$\{\{V\}(N_{\xi(n+1)})\}(N_{\xi(n)}) \left\{ \begin{array}{ll} \text{conv } N_1 & \text{if, in going from the } n\text{-th to the } (n+1)\text{-th} \\ & \text{complete configuration, the figure 0 is} \\ & \text{printed.} \\ \text{conv } N_2 & \text{if the figure 1 is printed.} \\ \text{conv } N_3 & \text{otherwise.} \end{array} \right.$$

Let W_γ stand for

$$\lambda u \left[\{\{V\}(\{A_\gamma\}(\{U_\gamma\}(u)))\}(\{U_\gamma\}(u)) \right],$$

so that, for each integer n ,

$$\{\{V\}(N_{\xi(n+1)})\}(N_{\xi(n)}) \text{ conv } \{W_\gamma\}(N_n),$$

and let Q be a formula such that

$$\{\{Q\}(W_\gamma)\}(N_s) \text{ conv } N_{r(s)},$$

where $r(s)$ is the s -th integer q for which $\{W_\gamma\}(N_q)$ is convertible into either N_1 or N_2 . Then, if M_γ stands for

$$\lambda w \left[\{W_\gamma\}(\{\{Q\}(W_\gamma)\}(w)) \right],$$

it will have the required property†.

The Graduate College,
Princeton University,
New Jersey, U.S.A.

† In a complete proof of the λ -definability of computable sequences it would be best to modify this method by replacing the numerical description of the complete configurations by a description which can be handled more easily with our apparatus. Let us choose certain integers to represent the symbols and the m -configurations of the machine. Suppose that in a certain complete configuration the numbers representing the successive symbols on the tape are $s_1 s_2 \dots s_n$, that the m -th symbol is scanned, and that the m -configuration has the number t ; then we may represent this complete configuration by the formula

$$[[N_{s_1}, N_{s_2}, \dots, N_{s_{m-1}}, [N_t, N_{s_m}], [N_{s_{m+1}}, \dots, N_{s_n}]]],$$

where

$$[a, b] \text{ stands for } \lambda u \left[\{\{u\}(a)\}(b) \right],$$

etc.

$$[a, b, c] \text{ stands for } \lambda u \left[\{\{\{u\}(a)\}(b)\}(c) \right],$$

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO
THE ENTSCHEIDUNGSPROBLEM. A CORRECTION

By A. M. TURING.

In a paper entitled "On computable numbers, with an application to the Entscheidungsproblem" * the author gave a proof of the insolubility of the Entscheidungsproblem of the "engere Funktionenkalkül". This proof contained some formal errors† which will be corrected here: there are also some other statements in the same paper which should be modified, although they are not actually false as they stand.

The expression for $\text{Inst}\{q_i S_j S_k Lq_l\}$ on p. 260 of the paper quoted should read

$$(x, y, x', y') \left\{ \left(R_{S_j}(x, y) \& I(x, y) \& K_{q_i}(x) \& F(x, x') \& F(y', y) \right) \right. \\ \rightarrow \left(I(x', y') \& R_{S_k}(x', y) \& K_{q_l}(x') \& F(y', z) \vee \left[\left(R_{S_0}(x, z) \rightarrow R_{S_0}(x', z) \right) \right. \right. \\ \left. \left. \& \left(R_{S_1}(x, z) \rightarrow R_{S_1}(x', z) \right) \& \dots \& \left(R_{S_M}(x, z) \rightarrow R_{S_M}(x', z) \right) \right] \right\},$$

S_0, S_1, \dots, S_M being the symbols which can print. The statement on p. 261, line 33, viz.

$$\text{"Inst}\{q_a S_b S_d Lq_c\} \& F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1})$$

is provable" is false (even with the new expression for $\text{Inst}\{q_a S_b S_d Lq_c\}$): we are unable for example to deduce $F^{(n+1)} \rightarrow (-F(u, u''))$ and therefore can never use the term

$$F(y', z) \vee \left[\left(R_{S_0}(x, z) \rightarrow R_{S_0}(x', z) \right) \& \dots \& \left(R_{S_M}(x, z) \rightarrow R_{S_M}(x', z) \right) \right]$$

* *Proc. London Math. Soc.* (2), 42 (1936-7), 230-265.

† The author is indebted to P. Bernays for pointing out these errors.

in $\text{Inst}\{q_a S_b S_d Lq_c\}$. To correct this we introduce a new functional variable G [$G(x, y)$ to have the interpretation “ x precedes y ”]. Then, if Q is an abbreviation for

$$(x)(\exists w)(y, z) \left[F(x, w) \& \left(F(x, y) \rightarrow G(x, y) \right) \& \left(F(x, z) \& G(z, y) \rightarrow G(x, y) \right) \right. \\ \left. \& \left[G(z, x) \vee \left(G(x, y) \& F(y, z) \right) \vee \left(F(x, y) \& F(z, y) \right) \rightarrow \left(\neg F(z, z) \right) \right] \right]$$

the corrected formula $\text{Un}(\mathfrak{U})$ is to be

$$(\exists u) A(\mathfrak{U}) \rightarrow (\exists s)(\exists t) R_{S_1}(s, t),$$

where $A(\mathfrak{U})$ is an abbreviation for

$$Q \& (y) R_{S_0}(u, y) \& I(u, u) \& K_{q_1}(u) \& \text{Des}(\mathfrak{U}).$$

The statement on page 261 (line 33) must then read

$$\text{Inst}\{q_a S_b S_d Lq_c\} \& Q \& F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1}),$$

and line 29 should read

$$r(n, i(n)) = b, \quad r(n+1, i(n)) = d, \quad k(n) = a, \quad k(n+1) = c.$$

For the words “logical sum” on p. 260, line 15, read “conjunction”. With these modifications the proof is correct. $\text{Un}(\mathfrak{U})$ may be put in the form (I) (p. 263) with $n = 4$.

Some difficulty arises from the particular manner in which “computable number” was defined (p. 233). If the computable numbers are to satisfy intuitive requirements we should have :

If we can give a rule which associates with each positive integer n two rationals a_n, b_n satisfying $a_n \leq a_{n+1} < b_{n+1} \leq b_n$, $b_n - a_n < 2^{-n}$, then there is a computable number a for which $a_n \leq a \leq b_n$ each n . (A)

A proof of this may be given, valid by ordinary mathematical standards, but involving an application of the principle of excluded middle. On the other hand the following is false :

There is a rule whereby, given the rule of formation of the sequences a_n, b_n in (A) we can obtain a D.N. for a machine to compute a . (B)

That (B) is false, at least if we adopt the convention that the decimals of numbers of the form $m/2^n$ shall always terminate with zeros, can be seen in this way. Let \mathfrak{U} be some machine, and define c_n as follows : $c_n = \frac{1}{2}$ if \mathfrak{U} has not printed a figure 0 by the time the n -th complete configuration is reached $c_n = \frac{1}{2} - 2^{-m-3}$ if 0 had first been printed at the m -th

complete configuration ($m \leq n$). Put $a_n = c_n - 2^{-n-2}$, $b_n = c_n + 2^{-n-2}$. Then the inequalities of (A) are satisfied, and the first figure of α is 0 if \mathfrak{N} ever prints 0 and is 1 otherwise. If (B) were true we should have a means of finding the first figure of α given the D.N. of \mathfrak{N} : *i.e.* we should be able to determine whether \mathfrak{N} ever prints 0, contrary to the results of § 8 of the paper quoted. Thus although (A) shows that there must be machines which compute the Euler constant (for example) we cannot at present describe any such machine, for we do not yet know whether the Euler constant is of the form $m/2^n$.

This disagreeable situation can be avoided by modifying the manner in which computable numbers are associated with computable sequences, the totality of computable numbers being left unaltered. It may be done in many ways* of which this is an example. Suppose that the first figure of a computable sequence γ is i and that this is followed by 1 repeated n times, then by 0 and finally by the sequence whose r -th figure is c_r ; then the sequence γ is to correspond to the real number

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

If the machine which computes γ is regarded as computing also this real number then (B) holds. The uniqueness of representation of real numbers by sequences of figures is now lost, but this is of little theoretical importance, since the D.N.'s are not unique in any case.

The Graduate College,
Princeton, N.J., U.S.A.

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

Computability and λ -definability

(*J. Symb. Log.* vol. **2** (1937), p. 153–163)

The p -function in λ - K conversion

(*J. Symb. Log.* vol. **2** (1937), p. 164)

PREFACE

These two papers appeared together and are connected, so it seems appropriate to review them together. They are both of a rather technical nature but the second is of comparatively minor significance.

For the purpose of stating the results below, a function is understood to be a *total* function from the set of natural numbers into itself, and is *computable* if it is computable by a Turing machine. Also, the term recursive here means *general* recursive. The results can be extended with little fuss to partial functions and partial recursiveness; this was first done by Kleene [1938].

The results of the first paper are that

- (i) every λ -definable function is computable.
- (ii) every computable function is recursive.

Kleene [1936] had already proved that every recursive function is λ -definable, and so these results completed the equivalence of the three concepts – computable, recursive and λ -definable. That is the first paper's historical contribution. It helped to cement opinion that a fundamental new concept had emerged in several different forms. See Gandy [1988].

Exposition of all these concepts has much improved but is still comparatively long and messy, and since it is now a standard part of recursion theory it would not be appropriate to repeat that level of detail here.

In order to prove (i), Turing actually proves (i)': every λ - K -definable function is computable. λ - K -definability was introduced by Kleene in [1936, p. 340–353] and is more general than λ -definability – when building up formulae, the scope of λx does not have to contain x free. So every λ -definable function is trivially λ - K -definable. It follows from (i)', (ii) and Kleene's result that every λ - K -definable function is λ -definable, so these two concepts are equivalent (already proved by Kleene).

The second of Turing's papers directly exhibits a λ - K -definable function which assigns to any positive integer n the least integer not less than n which has a given property. Kleene [1934] had already created a λ -definable function for this purpose, but Turing's example is simpler because it is allowed clauses unavailable to λ -definability.

This Page Intentionally Left Blank

COMPUTABILITY AND λ -DEFINABILITY

A. M. TURING

Several definitions have been given to express an exact meaning corresponding to the intuitive idea of 'effective calculability' as applied for instance to functions of positive integers. The purpose of the present paper is to show that the computable¹ functions introduced by the author are identical with the λ -definable² functions of Church and the general recursive³ functions due to Herbrand and Gödel and developed by Kleene. It is shown that every λ -definable function is computable and that every computable function is general recursive. There is a modified form of λ -definability, known as λ -K-definability, and it turns out to be natural to put the proof that every λ -definable function is computable in the form of a proof that every λ -K-definable function is computable; that every λ -definable function is λ -K-definable is trivial. If these results are taken in conjunction with an already available⁴ proof that every general recursive function is λ -definable we shall have the required equivalence of computability with λ -definability and incidentally a new proof of the equivalence of λ -definability and λ -K-definability.

A definition of what is meant by a computable function cannot be given satisfactorily in a short space. I therefore refer the reader to *Computable* pp. 230–235 and p. 254. The proof that computability implies recursiveness requires no more knowledge of computable functions than the ideas underlying the definition: the technical details are recalled in §5. On the other hand in proving that the λ -K-definable functions are computable it is assumed that the reader is familiar with the methods of *Computable* pp. 235–239.

The identification of 'effectively calculable' functions with computable functions is possibly more convincing than an identification with the λ -definable or general recursive functions. For those who take this view the formal proof of equivalence provides a justification for Church's calculus, and allows the 'machines' which generate computable functions to be replaced by the more convenient λ -definitions.

Received September 11, 1937.

¹ A. M. Turing, *On computable numbers, with an application to the Entscheidungsproblem*, *Proceedings of the London Mathematical Society*, ser. 2, vol. 42 (1936–7), pp. 230–265, quoted here as *Computable*. A similar definition was given by E. L. Post, *Finite combinatory processes—formulation 1*, this JOURNAL, vol. 1 (1936), pp. 103–105.

² Alonzo Church, *An unsolvable problem of elementary number theory*, *American journal of mathematics*, vol. 58 (1936), pp. 345–363, quoted here as *Unsolvable*.

³ S. C. Kleene, *General recursive functions of natural numbers*, *Mathematische Annalen*, vol. 112 (1935–6), pp. 727–742. A definition of general recursiveness is also to be found in *Unsolvable* pp. 350–351.

⁴ S. C. Kleene, λ -definability and recursiveness, *Duke mathematical journal*, vol. 2 (1936), pp. 340–353.

1. Definition of λ -K-definability. In this section the notion of λ -K-definability is introduced in a form suitable for handling with machines. There will be three differences from the normal, in addition to that which distinguishes λ -K-definability from λ -definability. One consists in using only one kind [] of bracket instead of three, { }, (), []; another is that x , x^1 , x^{11} , . . . are used as variables instead of an indefinite infinite list of the single symbols, and the third is a change in the form of condition (ii) of immediate transformability, not affecting the definition of convertibility except in form.

There are five symbols which occur in the formulae of the conversion calculus. They are λ , x , 1 , [and]. A sequence of symbols consisting of x followed by 1 repeated any number (possibly 0) of times is called a *variable*. *Properly-formed formulae* are a class of sequences of symbols which includes all variables. Also if M and N are⁵ properly-formed formulae, then $[M][N]$ (i.e. the sequence consisting of [followed by M then by], [and the sequence N , and finally by]) is a properly-formed formula. If M is a properly-formed formula and V is a variable, then $\lambda V[M]$ is a properly-formed formula. If any sequence is a properly-formed formula it must follow that it is so from what has already been said.

A properly-formed formula M will be said to be *immediately transformable* into N if either:

(i) M is of the form $\lambda V[X]$ and N is $\lambda U[Y]$ where Y is obtained from X by replacing the variable V by the variable U throughout, provided that U does not occur in X .

(ii) M is of the form $[\lambda V[X]][Y]$ where V is a variable and N is obtained by substituting Y for V throughout X . This is to be subject to the restriction that if W be either V or a variable occurring in Y , then λW must not occur in X .

(iii) N is immediately transformable into M by (ii).

A will be said to be immediately convertible into B if A is immediately transformable into B or if A is of the form $X[L]Y$ and B is $X[M]Y$ where L is immediately transformable into M . Either X or Y may be void. A is *convertible* to B (A conv B) if there is a finite sequence of properly-formed formulae, beginning with A and terminating with B , each immediately convertible into the preceding.

The formulae,

$$\begin{aligned} \lambda x[\lambda x^1[x^1]] & \quad (\text{abbreviated to } 0), \\ \lambda x[\lambda x^1[[x][x^1]]] & \quad (\text{abbreviated to } 1), \\ \lambda x[\lambda x^1[[x][[x][x^1]]]] & \quad (\text{abbreviated to } 2), \text{ etc.}, \end{aligned}$$

represent the natural numbers. If n represents a natural number then the next natural number is represented by a formula which is convertible to $[S][n]$ where S is

$$\lambda x^{11}[\lambda x[\lambda x^1[[x][[[x^{11}][x]][x^1]]]]].$$

A function $f(n)$ of the natural numbers, taking natural numbers as values will be said to be λ -K-definable if there is a formula F such that $[F][n]$ is con-

⁵ Heavy type capitals are used to stand for variable or undetermined sequences of symbols. In expressions involving brackets and heavy type letters it is to be understood that the possible substitutions of sequences of symbols for these letters is to be subject to the restriction that the pairing of the explicitly shown brackets is unaltered by the substitution; thus in $X[L]Y$ the number of occurrences of [in L must equal the number of occurrences of].

vertible to the formula representing $f(n)$ if n is the formula representing n . The formula $[F][n]$ can never be convertible to two formulae representing different natural numbers, for it has been shown⁶ that if two properly-formed formulae are in normal form (i.e., have no parts of the form $[\lambda V[M]][N]$) and are convertible into one another, then the conversion can be carried out by the use of (i) only. The formulae representing the natural numbers are in normal form and the formulae representing two different natural numbers are certainly not convertible into one another by the use of (i) alone.

2. Abbreviations. A number of abbreviations of the same character as those in *Computable* (pp. 235–239) are introduced here. They will be applied in connection with the calculus of conversion, but are necessary for other purposes, e.g. for carrying out the processes of any ordinary formal logic with machines. The abbreviations in *Computable* are taken as known.

‘The sequence of symbols marked with α (followed by α)’ will be abbreviated to $S(\alpha)$ in the explanations. Sequences are normally identified by the way they are marked, and are as it were lost when their marks are erased.

In the tables α will be used as a name for the symbol ‘blank.’

$\text{pem}(\mathfrak{A}, \alpha, \beta)$		$\text{pe}(\text{pem}_1, \alpha)$
pem_1	$R, P\beta$	\mathfrak{A}
pem_1 here stands for $\text{pem}_1(\mathfrak{A}, \alpha, \beta)$ and similar abbreviations must be understood throughout.		
$\text{pem}(\mathfrak{A}, \alpha, \beta)$. The machine prints α at the end of the sequence of symbols on F-squares and marks it with β . $\rightarrow \mathfrak{A}$.		

The tables for $\text{crm}(\mathfrak{B}, \gamma, \beta)$ and $\text{cem}(\mathfrak{B}, \gamma, \beta)$ are to be obtained from those for $\text{cr}(\mathfrak{B}, \gamma)$ and $\text{ce}(\mathfrak{B}, \gamma)$ by replacing $\text{pe}(\mathfrak{A}, \alpha)$ by $\text{pem}(\mathfrak{A}, \alpha, \beta)$ throughout.

$\text{cpr}(\mathfrak{A}, \mathfrak{E}, \alpha, \beta)$		$\text{cp}(\text{cpr}_1, \text{cpr}_2, \text{cpr}_3, \alpha, \beta)$
cpr_1		$\text{re}(\text{re}(\text{cpr}, b, \beta, b), b, \alpha, a)$
cpr_2		$\text{re}(\text{re}(\mathfrak{E}, b, \beta), a, \alpha)$
cpr_3		$\text{re}(\text{re}(\mathfrak{A}, b, \beta), a, \alpha)$

$\text{cpr}(\mathfrak{A}, \mathfrak{E}, \alpha, \beta)$. The machine compares $S(\alpha)$ with $S(\beta)$. $\rightarrow \mathfrak{A}$ if they are alike; $\rightarrow \mathfrak{E}$ otherwise. No erasures are made.

The letters a, b occurring in the table for cpr should not be used elsewhere in any machine whose table involves cpr . This can be made automatic by using a_{cpr} and b_{cpr} , say, instead of a and b . We shall however write a and b and understand them to mean a_{cpr} and b_{cpr} . The same applies for the letters a, \dots, z in all such tables.

$\text{f}(\mathfrak{A}, \gamma)$	$\begin{cases} \gamma \\ \text{not } \gamma \end{cases}$	L	\mathfrak{A}
$\text{bf}(\mathfrak{A}, \alpha, \beta, \gamma, \delta)$		R, R	\mathfrak{f}
		E, Pa	$\text{f}(\text{bf}_1, \gamma)$

⁶ Alonzo Church and J. B. Rosser, *Some properties of conversion*, *Transactions of the American Mathematical Society*, vol. 39 (1936), pp. 472–482. The result used here is Theorem 1 Corollary 2 as extended to the modified conversion on p. 482.

bf_1	$\begin{cases} \alpha \\ \beta \\ \text{others} \end{cases}$	R, E, Pb	$f'(bf_2, \gamma)$
		L	bf_3
		R, R, R	$f'(bf_1, \gamma)$
bf_2	$\begin{cases} \beta \\ \text{not } \beta \end{cases}$	R, E, Pb	$f'(bf_1, \beta)$
	$\begin{cases} \gamma \text{ or } b \\ \alpha \end{cases}$	R, R, R	$f'(bf_2, \gamma)$
bf_3	$\begin{cases} \alpha \\ \text{others} \end{cases}$	$E, P\delta, L, L$	bf_3
		$E, P\gamma$	\mathfrak{A}
		L, L	bf_3

$bf(\mathfrak{A}, \alpha, \beta, \gamma, \delta)$. This describes the process of finding the partner of a bracket. If α and β are regarded as left and right brackets, then if the machine takes up the internal configuration bf when scanning a square next on the right of an α it will find the partner of this α in the sequence $S(\gamma)$, and will mark the part of $S(\gamma)$ which is between the brackets with δ (instead of γ). The final internal configuration is \mathfrak{A} and the scanned square is that which was scanned when the internal configuration bf was first taken up.

$sb(\mathfrak{A}, \alpha, \beta, \gamma, \delta, \epsilon)$		$f'(sb_1, \text{crm}(\text{re}(\text{re}(sb, a, j), b, \beta), \gamma, \epsilon), \beta)$
sb_1	σ	R, E, Pb
		$sb_2(\mathfrak{A}, \alpha, \beta, \gamma, \delta, \epsilon, \sigma)$
		$f'(sb_3, \text{crm}(\text{re}(\text{re}(sb, b, \beta), j, \alpha), a, \alpha), a, \delta), \alpha)$
sb_3	$\begin{cases} \sigma \\ \text{not } \sigma \end{cases}$	R, E, Pa
		sb
	R, E, Pa	$re(f'(sb_4, b, a), b, \beta)$
sb_4	τ	R, E, Pj
		$re(pem(sb, \tau, \delta), a, \alpha)$
$sub(\mathfrak{A}, \alpha, \beta, \gamma, \delta)$		$sb(\mathfrak{A}, \alpha, \beta, \gamma, \delta, \delta)$
$dt(\mathfrak{A}, \mathfrak{B}, \alpha, \beta)$		$pem(sb(f(e(\mathfrak{A}, d), \mathfrak{B}, d), \alpha, \beta, p, \mathfrak{a}, d), r, p)$

$sub(\mathfrak{A}, \alpha, \beta, \gamma, \delta)$. $S(\gamma)$ is substituted for $S(\beta)$ throughout $S(\alpha)$. The result is copied down and marked with δ . $\rightarrow \mathfrak{A}$.

$dt(\mathfrak{A}, \mathfrak{B}, \alpha, \beta)$. It is determined whether the sequence $S(\beta)$ occurs in $S(\alpha)$. $\rightarrow \mathfrak{A}$ if it does; $\rightarrow \mathfrak{B}$ otherwise.

The tables which follow are particularly important in all cases where an enumeration of all possible results of operations of given types is required. The enumeration may be carried out by regarding the operation as determined by a number of choices, each between two possibilities, L and M say. Each possible sequence of operations is then associated with a finite sequence of letters L and M. These sequences can easily be enumerated. The method used here is to replace L by 0, each M by 1, follow the whole by 1, reverse the order and regard the result as the binary Arabic numeral corresponding to the given sequence. Thus the first few sequences (beginning with the one associated with 1) are: the null sequence, L, M, LL, ML, LM, MM, LLL, MLL, LML, MML. In the general table below ξ and η are used instead of L and M.

$abb(\mathfrak{A}, \alpha, \xi, \eta)$		$f'(abb_1, pem(abb_2, \xi, a), \alpha)$
abb_1	$\begin{cases} \eta \\ \xi \end{cases}$	R, E
		$pem(abb, \xi, a)$
abb_2		$pem(abb_2, \eta, a)$
		$cem(re(\mathfrak{A}, a, \alpha), \alpha, a)$

$\text{add}(\mathfrak{A}, \alpha, \xi, \eta)$. The sequence $S(\alpha)$ consisting of letters ξ and η only is transformed into the next sequence. $\rightarrow \mathfrak{A}$.

$\text{ch}(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \alpha, \xi, \eta)$		$f'(\text{ch}_1, \text{re}(\mathfrak{C}, b, \alpha), \alpha)$
ch_1	$\begin{cases} \xi \\ \eta \end{cases}$	R, E, Pb
		\mathfrak{A}
		R, E, Pb
		\mathfrak{B}

$\text{ch}(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \alpha, \xi, \eta)$ is an internal configuration which is taken up when a choice has to be made. $S(\alpha)$ is the sequence of letters ξ and η determining the choices. $\rightarrow \mathfrak{A}$ if the first unused letter is ξ ; $\rightarrow \mathfrak{B}$ if it is η : it is then indicated that this ξ or η has been used by replacing its mark by b . When the whole sequence has been used up these marks are replaced by α again and $\rightarrow \mathfrak{C}$.

$\text{cch}(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \alpha, \xi, \eta)$	R	cch_1
cch_1	σ	E, Pa
cch_2		$\text{cch}_2(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \alpha, \xi, \eta, \sigma)$
cch_3		$\text{ch}(\text{f}(\text{cch}_3, b, a), \text{f}(\text{cch}_4, b, a), \mathfrak{C}, \alpha, \xi, \eta)$
cch_4		\mathfrak{A}
		$E, P\sigma, L$
		\mathfrak{B}
		$E, P\sigma, L$

$\text{cch}(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \alpha, \xi, \eta)$. This differs from ch in that the internal configurations \mathfrak{A} and \mathfrak{B} are taken up when the same square is scanned as that which was scanned when the internal configuration cch was first reached, provided that this was an F-square.

3. Mechanical Conversion. We are now in a position to show how the conversion process can be carried out mechanically. It will be necessary to be able to perform all of the three kinds of immediate transformation. (iii) can be done most easily if we can enumerate properly-formed formulae. It is principally for this purpose that we introduce the table for $\text{pff}(\mathfrak{A}, \alpha)$.

$\text{funf}(\mathfrak{A}, \alpha, \beta, \gamma) \quad \text{pem}(\text{crm}(\text{pem}_2(\text{crm}(\text{pem}(\mathfrak{A},], \gamma), \beta, \gamma),], [\gamma), \alpha, \gamma), [\gamma)$
 $\text{funf}(\mathfrak{A}, \alpha, \beta, \gamma) \cdot [S(\alpha)] [S(\beta)]$ is written at the end and marked with γ . $\rightarrow \mathfrak{A}$.

$\text{ch}(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \theta)$	$\text{ch}(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \theta, L, M)$
$\text{cch}(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \theta)$	$\text{cch}(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \theta, L, M)$

The choices will be determined by a sequence made up of letters L and M .

$\text{pff}(\mathfrak{A}, \mathfrak{C}, \alpha, \theta)$		$\text{pff}_5(\text{ch}(\text{af}, \text{pff}_1, \mathfrak{C}, \theta), :, ;, x, ;, x)$
pff_1		$\text{q}(\text{pff}_2, :)$
pff_2	$\begin{cases} ; \\ \alpha \\ \text{others} \end{cases}$	$\text{ch}(\text{pff}_3, \text{pff}_2, \mathfrak{C}, \theta)$
	R, R	af
	R, R	pff_2
pff_3	$\begin{cases} ; \\ \alpha \\ \text{others} \end{cases}$	pff_4
	R, Pa, R	af
	R, R	pff_3
pff_4	$\begin{cases} ; \\ \alpha \\ \text{others} \end{cases}$	$\text{ch}(\text{pff}_5, \text{pff}_4, \mathfrak{C}, \theta)$
	R, R	af
	R, R	pff_4
pff_5	$\begin{cases} ; \text{ or } \alpha \\ \text{others} \end{cases}$	ar
	R, Pb, R	pff_5

or		ch(ne, ch(comp, ab, C, θ), C, θ)	
ne		pe ₂ (ne ₁ , ;, x)	
ne ₁		ch(pe(ne ₁ , !), af, C, θ)	
comp		pe(funk(af, a, b, a), ;)	
ab		pe ₃ (ab ₁ , ;, λ, x)	
ab ₁		ch(pe(ab ₁ , !), ab ₂ , C, θ)	
ab ₂		pe(ce(pe(af, !), a), !)	
af		e(e(ch(fin, pff ₁ , C, θ), a), b)	
fin		q(r(r(fin ₁)), ;)	
fin ₁	$\begin{cases} \text{not } a \\ \text{others} \end{cases}$	$\begin{cases} R, Pa, R \\ \mathfrak{A} \end{cases}$	$\begin{cases} fin_1 \\ \mathfrak{A} \end{cases}$

$\text{pff}(\mathfrak{A}, C, \alpha, \theta)$. A properly-formed formula is chosen, written down at the end and marked with α . $\rightarrow \mathfrak{A}$. This is done by writing down successive properly-formed formulae separated by semicolons, and obtaining others from them by abstraction (i.e., the process by which $\lambda V[M]$ is obtained from M), by application of a function to its argument (i.e., obtaining $[M][N]$ from M and N), and by writing down new variables. Before writing down a new formula we have the alternative of taking the last formula as the result of the calculation. In this case the internal configuration fin is taken up. If a new formula is to be constructed then two of the old formulae are chosen and marked with a and b : then one of the internal configurations ab , comp , ne is chosen and the new formula is correspondingly $\lambda V[S(a)]$, $[S(a)][S(b)]$, or V , where V is a new variable. The whole of the work is separated by a colon from the symbols which were on the tape previously. The meanings of pe_3 and pe_5 are analogous to pe_2 .

The occurrence of λ in this table is of course as a symbol of the conversion calculus, not as a variable machine symbol.

The immediate transformations (i) and (ii) are described next.

va($\mathfrak{A}, C, \alpha, \beta, \theta$)		$f'(va_1, \mathfrak{A}, \alpha)$	
va ₁	$\begin{cases} \lambda \\ \text{others} \end{cases}$	$\begin{cases} R, E, Pa \\ \mathfrak{A} \end{cases}$	$\begin{cases} f'(va_2, b, \alpha) \\ \text{crm}(\mathfrak{A}, \alpha, \beta) \end{cases}$
va ₂	$\begin{cases} x \text{ or } ! \\ \text{others} \end{cases}$	$\begin{cases} R, E, Pb \\ R \end{cases}$	$\begin{cases} f'(va_2, b, \alpha) \\ \text{bf}(pe(va_3, x), [], \alpha, c) \end{cases}$
va ₃		R, Pd	$ch(pe(va_3, !), dt(va_4, va_5, c, d), C, \theta)$
va ₄			$re(re(re(crm(e(\mathfrak{A}, d), \alpha, \beta), b, \alpha), a, \alpha), c, \alpha)$
va ₅			$crm(re(re(va_6, a, \alpha), b, \alpha), c, \alpha), b, f)$
va ₆			$sub(e(e(\mathfrak{A}, d), f), \alpha, f, d, \beta)$

$va(\mathfrak{A}, C, \alpha, \beta, \theta)$. An immediate transformation (i) is chosen, and if permissible is carried out on $S(\alpha)$, the result being marked with β . If the chosen transformation is not permissible then $S(\beta)$ is identical with $S(\alpha)$. $\rightarrow \mathfrak{A}$.

$red(\mathfrak{A}, \alpha, \beta)$

red_1	$\begin{cases} [\\ \text{not } [\end{cases}$	R	$\text{bf}(\text{red}_2, [,], \alpha, c)$
red_2		E, Pf	red_{13}
red_3			$\text{re}(\text{f}(\text{red}_3, b, \alpha), b, \alpha, f)$
red_4			$\text{bf}(\text{red}_4, [,], \alpha, d)$
red_5	$\begin{cases} \lambda \\ \text{not } \lambda \end{cases}$	R, E, Pf	$\text{f}'(\text{red}_5, b, c)$
red_6	$\begin{cases} x \text{ or } ! \\ [\end{cases}$	R, E, Pg	$\text{f}'(\text{red}_6, b, c)$
red_7	$[$	R, E, Pf	$q(\text{red}_7, c)$
red_8		E, Pf	red_8
red_9	$\begin{cases} \lambda \\ \text{not } \lambda \end{cases}$	R, E, Pk	$\text{f}'(\text{red}_9, \text{red}_{10}, c)$
red_{11}	$\begin{cases} x \text{ or } ! \\ [\end{cases}$	R, E, Pj	$\text{f}'(\text{red}_{11}, b, c)$
red_{12}			$\text{cpr}(\text{red}_{13}, \text{red}_{12}, j, g)$
red_{10}			$\text{dt}(\text{red}_{13}, \text{re}(\text{red}_8, j, k), d, j)$
red_{13}		$\text{sub}(\text{re}(\text{re}(\text{re}(\text{re}(\text{re}(\mathfrak{A}, d, \alpha), f, \alpha), k, \alpha), g, \alpha)k, g, d, \beta)$	$\text{re}(\text{re}(\text{re}(\text{re}(\text{re}(\text{re}(\text{crm}(\mathfrak{A}, \alpha, \beta), d, \alpha), g, \alpha), c, \alpha), f, \alpha), k, \alpha), j, \alpha)$

$\text{red}(\mathfrak{A}, \alpha, \beta)$. An immediate transformation (ii) is carried out on $S(\alpha)$, supposing that $S(\alpha)$ is properly-formed. The result is marked with $\beta \rightarrow \mathfrak{A}$. If the transformation is not possible or permissible $S(\beta)$ is identical with $S(\alpha)$. Considerable use is made of the hypothesis that $S(\alpha)$ is properly-formed. Thus if its first symbol is $[$ then it must be of form $[L][N]$ and if in addition the second symbol is λ then it is of form $[\lambda V[M]][N]$. The internal configuration red_8 is never reached unless $S(\alpha)$ is of this form, and in that case it first occurs when V has been marked with g , M with c , and N with d , the remaining symbols of what was $S(\alpha)$ being now marked with α or f . It is then determined whether the immediate transformation (ii) is permissible: if it is then red_{10} is taken up and the substitution carried out.

$\text{imc}(\mathfrak{A}, \mathfrak{C}, \alpha, \beta, \theta)$		$\text{ch}(\text{f}'(\text{imc}_1, \text{imc}_2, \alpha), \text{re}(\text{imc}_4, \alpha, a), \mathfrak{C}, \theta)$	
imc_1	$\begin{cases} [\\ \text{not } [\end{cases}$	R, E, Pc	$\text{ch}(\text{imc}, \text{imc}_3, \mathfrak{C}, \theta)$
imc_2	$[$		imc
imc_3			$\text{re}(\text{crm}(\mathfrak{A}, \alpha, \beta), c, \alpha)$
imc_4			$\text{q}(\text{bf}(\text{imc}_4, [,], \alpha, a), c)$
vc			$\text{ch}(\text{vc}, \text{ch}(\text{rc}, \text{ex}, \mathfrak{C}, \theta), \mathfrak{C}, \theta)$
rc			$\text{va}(\text{imc}_5, \mathfrak{C}, a, b, \theta)$
ex			$\text{red}(\text{imc}_5, a, b)$
			$\text{pff}(\text{red}(\text{ex}_1, b, d), \mathfrak{C}, b, \theta)$

ex_1 $\text{cpr}(\text{e}(\text{imc}_5, d), \text{e}(\text{e}(\text{crm}(\text{imc}_5, a, b), d), b), d, a)$
 imc_5 $\text{crm}(\text{crm}(\text{crm}(\text{re}(\text{re}(\text{e}(\mathfrak{A}, b), c, \alpha), a, \alpha), \alpha, \beta), b, \beta), c, \beta)$

$\text{imc}(\mathfrak{A}, \mathfrak{C}, \alpha, \beta, \theta)$. An immediate conversion is chosen and performed on $S(\alpha)$. The result is marked with β . $\rightarrow \mathfrak{A}$.

$\text{conv}(\mathfrak{A}, \alpha, \beta, \theta)$	$\text{pe}(\text{crm}(\text{conv}_1, \alpha, d), .)$
conv_1	$\text{ch}(\text{imc}(\text{conv}_2, \text{au}, d, f, \theta), \text{re}(\mathfrak{A}, d, \beta), \text{au}, \theta)$
conv_2	$\text{e}(\text{re}(\text{conv}_1, f, d), d)$
au	$\text{q}(\text{au}_1, .)$
au_1	$\left\{ \begin{array}{ll} \mathfrak{a} & \text{crm}(\mathfrak{A}, \alpha, \beta) \\ \text{not } \mathfrak{a} & \text{R, E, R} \end{array} \right.$ au_1

$\text{conv}(\mathfrak{A}, \alpha, \beta, \theta)$. A conversion is chosen and performed on $S(\alpha)$. The result is marked with β . $\rightarrow \mathfrak{A}$. The sequence determining the choices is $S(\theta)$. If it should happen that this sequence is exhausted before the conversion is completed then the final formula is the same as the original, i.e. $S(\alpha)$. The half finished conversion work is effectively removed from the tape by erasing the marks.

4. Computability of λ -K-definable functions. It is now comparatively simple to show that a λ -K-definable function is computable, i.e., that⁷ if $f(n)$ is λ -K-definable then the sequence γ_f in which there are $f(n)$ figures 1 between the n th and the $(n+1)$ th 0, and $f(0)$ figures before the first 0, is computable.

To simplify the table for the machine which computes γ_f we use the abbreviation $\text{Wr}(\mathfrak{A}, \mathbf{M}, \alpha)$ for an internal configuration starting from which the machine writes the sequence \mathbf{M} of symbols at the end, marking it with α and finishing in the internal configuration \mathfrak{A} . Thus the table for $\text{Wr}(\mathfrak{A}, \lambda x^1, \alpha)$ would be:

$\text{Wr}(\mathfrak{A}, \lambda x^1, \alpha)$	$\text{pe}(\text{Wr}_1, \mathfrak{a})$
Wr_1	$P\lambda, R, P\alpha, R, Px, R, P\alpha, R, P^1, R, P\alpha$

We use one more skeleton table:

$\text{pls}(\mathfrak{A}, \alpha, \beta)$	$\text{funf}(\text{e}(\text{re}(\mathfrak{A}, a, \alpha), \alpha), \beta, \alpha, a)$
---	---

If F is the formula which λ -K-defines $f(n)$ then the table for the machine which computes γ_f is:

\mathfrak{b}	$P\vartheta, R, P\vartheta$	$\text{Wr}(\mathfrak{b}_1, F, h)$
\mathfrak{b}_1		$\text{Wr}(\mathfrak{b}_2, \lambda x^1 [x^1], i)$
\mathfrak{b}_2		$\text{crm}(\mathfrak{b}_3, i, k)$
\mathfrak{b}_3	$\text{Wr}(\mathfrak{b}_4, \lambda x^1 [x^1] [x^1] [x^1] [x^1] [x^1], u)$	$\text{funf}(\text{cn}_1, h, k, v)$
\mathfrak{b}_4		$\text{add}(\text{cn}_1, s, L, M)$
cn		$\text{crm}(\text{cn}_2, i, d)$
cn_1		$\text{ch}(\text{re}(\text{cn}_3, d, m), \text{pls}(\text{cn}_2, d, u), \text{cn}_6, s)$
cn_2		$\text{conv}(\text{cn}_4, v, w, s)$
cn_3		$\text{cpr}(\text{cn}_5, \text{cn}_6, w, m)$
cn_4		

⁷ *Computable* p. 254.

cn_6		$e(e(e(cn_{10}, w), m), d)$
cn_{10}		$ch(cn_{10}, cn_{10}, cn, s)$
cn_5		$q(cn_7, m)$
cn_7	E	$q(cn_8, m)$
cn_8	E	$q(l(cn_9), m)$
cn_9	R, E	$pem(q(l(cn_9), m), 1, a)$
ba_1	{ not]	$pem(e(e(e(e(ba_1, s), v), w), m), 0, a)$
		$pls(ba, k, u)$

When the machine reaches the internal configuration ba for the $(n+1)$ th time ($n \geq 0$) the tape bears the formula F marked with h , the formula n representing the natural number n (or rather a formula convertible into it) marked with k , 0 marked with i , and S marked with u . A formula convertible into one representing some natural number r is then chosen and marked with m . This brings us to the internal configuration cn_3 . A conversion is then chosen and performed on $S(v)$, i.e. on $[F][n]$. The result is marked with w and compared with $S(m)$. If they are not alike the letters w, m are erased and we go back to cn_1 after transforming the sequence $S(s)$ which determines the choices into the next sequence. If they are alike then 1 is written at the end repeated r times followed by 0, all of which is marked with a . In order to have the correct number of figures we make use of the fact that the number of brackets occurring consecutively at the end of $S(m)$ is $r+2$. The machine is back in the internal configuration ba as soon as $S(k)$ has been changed to $[S][S(k)]$.

No attempt is being made to give a formal proof that this machine has the properties claimed for it. Such a formal proof is not possible unless the ideas of substitution and so forth occurring in the definition of conversion are formally defined, and the best form of such a definition is possibly in terms of machines.

If $f(n)$ ($n \geq 1$) is λ -definable, i.e. if F is well-formed (*Unsolvable* p. 346), then the present argument shows also that $f(n)$ is then computable in the sense that a function $g(n)$ of positive integers is computable if there is a computable sequence with $g(n)$ figures 1 between the n th and $(n+1)$ th figure 0.

5. Recursiveness of computable functions. It will now be shown that every computable function $f(n)$ of the natural numbers, taking natural numbers as values, is general recursive. We shall in fact find primitive recursive functions $j(x), \phi(x)$ such that if $\xi(x)$ is the $(x+1)$ th ($x=0, 1, 2, \dots$) natural number y for which $j(y)=0$, then $f(x)$ is given by

$$f(x) = \phi(\xi(x)).$$

It is easily seen that such a function is general recursive (cf. *Unsolvable* p. 353); also it can easily be brought into the form,⁸

$$f(x) = \phi(\epsilon y[i(x, y) = 0])$$

⁸ This may be done by defining $i(x, y)$ as follows:

$$\begin{aligned} e(0) &= 0, \\ e(S(x)) &= S(e(x)) && \text{if } j(x) = 0, \\ &= e(x) && \text{otherwise,} \\ i(x, y) &= \text{Max } (0, x - e(y)), \end{aligned}$$

$S(x)$ as usual meaning $x+1$.

(where $\epsilon y[i(x, y) = 0]$ means 'the least natural number y for which $i(x, y) = 0$ ', and $i(x, y)$ is primitive recursive) which plays a central part⁹ in the theory of general recursive functions. It would be slightly simpler to set up recursion equations for $f(x)$ but in that case it would be necessary to show that they were consistent; this is avoided by confining ourselves to primitive recursions (whose consistency is not likely to be doubted) except at the step from $j(x)$ to $\xi(x)$.

We are given the description of a machine which computes $f(x)$. The machine writes down symbols on a tape: amongst these symbols occur figures 0 and 1. The number of figures 1 between the n th and the $(n+1)$ th figure 0 is $f(n)$. At any moment there is one of the symbols on the tape which is to be distinguished from the others and is called the 'scanned symbol.' The state (complete configuration) of the system at any moment is described by the sequence of symbols on the tape, with an indication as to which of them is scanned, and the internal configuration (m -configuration in *Computable*) of the machine. As names for the symbols we take S_0, S_1, \dots, S_{N-1} and for the internal configurations q_1, q_2, \dots, q_R . Certain of these are names of definite symbols and internal configurations independent of the machine; in fact,

S_0 always stands for 'blank,'

S_1 always stands for 0,

S_2 always stands for 1,

q_1 always stands for the initial internal configuration.

If at any time there is the sequence

$$S_{s_1}, S_{s_2}, \dots, S_{s_k}, \dots, S_{s_{k+1}} \quad (k > 0, l \geq 0)$$

of symbols on the tape, with the k th symbol scanned and the internal configuration q_i , this complete configuration may be described by the four numbers,

$$w = s_{k-1} + Ns_{k-2} + \dots + N^{k-2}s_1,$$

s_k, t , and

$$v = s_{k+1} + Ns_{k+2} + \dots + N^{l-1}s_{k+l}$$

or by the single number,

$$u = p(w, s_k, t, v),$$

where

$$p(x_1, x_2, x_3, x_4) = 2^{x_1}3^{x_2}5^{x_3}7^{x_4}.$$

Each complete configuration of the machine is determined by the preceding one. The manner of this determination is given in the description of the machine, which consists of a number of expressions each of one of the forms $q_i S_s S_{s'} L q_{i'}$, or $q_i S_s S_{s'} N q_{i'}$, or $q_i S_s S_{s'} R q_{i'}$. The occurrence of the first of these means that if in any complete configuration the scanned symbol was S_s and the internal configuration q_i , then the machine goes to the next complete configuration by replacing the scanned symbol by $S_{s'}$ and making the new scanned symbol the symbol on the left of it and the new internal configuration $q_{i'}$. In other words if a complete configuration be described by the number,

$$p(s_{k-1} + Ns_{k-2} + \dots + N^{k-2}s_1, s, t, s_{k+1} + Ns_{k+2} + \dots + N^{l-1}s_{k+l}) \\ = p(s_{k-1} + Nf, s, t, s_{k+1} + Ng),$$

⁹ Compare the two papers by Kleene already quoted.

and if $q_t S_s S_s' Lq_t'$ occurs in the description of the machine, then the number describing the next complete configuration is

$$p(f, s_{k-1}, t', s' + N(s_{k+1} + Ng)).$$

In the case where we have $q_t S_s S_s' Nq_t'$, the next complete configuration will be described by

$$p(s_{k-1} + Nf, s', t', s_{k+1} + Ng),$$

and in the case of $q_t S_s S_s' Rq_t'$ by

$$p(s' + N(s_{k-1} + Nf), s_{k+1}, t', g).$$

We may define a primitive recursive function $d_1(s, t)$ (or $d_2(s, t)$ or $d_3(s, t)$) to have the value 1 or 0 according as an expression of the form $q_t S_s S_s' Lq_t'$ (or $q_t S_s S_s' Nq_t'$ or $q_t S_s S_s' Rq_t'$) does or does not occur in the description of the machine. In each of the three cases $z(s, t)$ is to have the value s' and $c(s, t)$ to have the value t' . $q(x)$, $r(x)$ are to be respectively quotient and remainder of x on division by N , and $\omega_r(x)$ ($r=2, 3, 5, 7$) is to be the greatest integer k for which r^k divides x . These functions are primitive recursive.

Then if we put

$$\begin{aligned} \theta(x) = & d_1(\omega_3(x), \omega_5(x))p(q(\omega_2(x)), r(\omega_2(x)), c(\omega_3(x), \omega_5(x)), z(\omega_3(x), \omega_5(x)) + N\omega_7(x)) \\ & + d_2(\omega_3(x), \omega_5(x))p(\omega_2(x), z(\omega_3(x), \omega_5(x)), c(\omega_3(x), \omega_5(x)), \omega_7(x)) \\ & + d_3(\omega_3(x), \omega_5(x))p(z(\omega_3(x), \omega_5(x)) + N\omega_2(x), r(\omega_7(x)), c(\omega_3(x), \omega_7(x)), q(\omega_7(x))), \end{aligned}$$

and

$$\begin{aligned} u(0) &= p(0, 0, 1, 0) = 5, \\ u(S(x)) &= \theta(u(x)), \end{aligned}$$

$u(x)$ will be the number describing the $(x+1)$ th complete configuration of the machine.

$g(x, y)$ is to be defined by

$$\begin{aligned} g(S(x), y) &= 2 \quad (\text{all } x, y \geq 0), & g(0, 1) &= 0, \\ g(0, 0) &= 2, & g(0, 2) &= 1, \\ g(0, x) &= 2 \quad (x \geq 3), \end{aligned}$$

and $j(x)$ by,

$$j(x) = g(\omega_3(u(x)), z(\omega_3(u(x)), \omega_5(u(x)))).$$

Then $j(x)=0$ means that in going from the $(x+1)$ th to the $(x+2)$ th complete configuration the machine prints a figure 0: if $j(x)=1$ it prints 1: $j(x)=2$ otherwise. $\xi(x)$ is defined to be the $(x+1)$ th natural number y for which $j(y)=0$, and $\phi(x)$ as follows:

$$\begin{aligned} \phi(0) &= 0, \\ \phi(S(x)) &= 0 \quad \text{if } j(x) = 0, \\ &= \phi(x) \quad \text{if } j(x) = 2, \\ &= S(\phi(x)) \quad \text{if } j(x) = 1. \end{aligned}$$

Then $\phi(x)$ is the number of times 1 has been printed since the last 0, reckoned at the $(x+1)$ th complete configuration. $\phi(\xi(x))$ is the number of times 1 occurs between the x th and the $(x+1)$ th figure 0, its value when $x=0$ being the number of figures 1 which precede all figures 0. But these are the properties which define $f(x)$.

THE \mathfrak{p} -FUNCTION IN λ - K -CONVERSION

A. M. TURING

In the theory of conversion it is important to have a formally defined function which assigns to any positive integer n the least integer not less than n which has a given property. The definition of such a formula is somewhat involved.¹ I propose to give the corresponding formula in λ - K -conversion,² which will (naturally) be much simpler. I shall in fact find a formula \mathfrak{p} such that if T be a formula for which $T(n)$ is convertible³ to a formula representing a natural number, whenever n represents a natural number, then $\mathfrak{p}(T, r)$ is convertible to the formula q representing the least natural number q , not less than r , for which $T(q)$ conv 0.² The method depends on finding a formula Θ with the property that Θ conv $\lambda u. u(\Theta(u))$, and consequently if $M \rightarrow \Theta(V)$ then M conv $V(M)$. A formula with this property is,

$$\Theta \rightarrow \{\lambda vu. u(v(v, u))\}(\lambda vu. u(v(v, u))).$$

The formula \mathfrak{p} will have the required property if $\mathfrak{p}(T, r)$ conv r when $T(r)$ conv 0, and $\mathfrak{p}(T, r)$ conv $\mathfrak{p}(T, S(r))$ otherwise. These conditions will be satisfied if $\mathfrak{p}(T, r)$ conv $T(r, \lambda x. \mathfrak{p}(T, S(r)), r)$, i.e. if \mathfrak{p} conv $\{\lambda p t r. t(r, \lambda x. \mathfrak{p}(t, S(r)), r)\}(\mathfrak{p})$. We therefore put,

$$\mathfrak{p} \rightarrow \Theta(\lambda p t r. t(r, \lambda x. \mathfrak{p}(t, S(r)), r)).$$

This enables us to define also a formula,

$$\mathcal{P} \rightarrow \lambda t n. n(\lambda v. \mathfrak{p}(t, S(v)), 0),$$

such that $\mathcal{P}(T, n)$ is convertible to the formula representing the n th positive integer q for which $T(q)$ conv 0.

PRINCETON UNIVERSITY

Received April 23, 1937.

¹ Such a function was first defined by S. C. Kleene, *A theory of positive integers in formal logic*, *American journal of mathematics*, vol. 57 (1934), see p. 231.

² For the definition of λ - K -conversion see S. C. Kleene, *λ -definability and recursiveness*, *Duke mathematical journal*, vol. 2 (1936), pp. 340–353, footnote 12. In λ - K -conversion we are able to define the formula $0 \rightarrow \lambda f x. x$. The same paper of Kleene contains the definition of a formula L with a property similar to the essential property of Θ (p. 346).

³ “Convertible” and “conv” refer to λ - K -conversion throughout this note.

Systems of logic based on ordinals

(Proc. Lond. Math. Soc., series 2 vol. **45** (1939), pp. 161–228)

This paper deserves to be read and understood far more than it has been. For an early estimation of the paper, the reader is again referred to Newman's obituary to Turing reproduced in this volume. It was for many years regarded as a difficult piece of work, and it was only when Feferman translated the work from the λ -calculus into the language of conventional recursive function theory (around twenty years later) that it began to be properly evaluated. Even now, it is still not *widely* understood and appreciated.

The preface which follows has been adapted from Feferman's excellent paper [1988]: 'Turing in the Land of $O(z)$ '. The editor wishes to reiterate his gratitude to both Professor Feferman and Oxford University Press, the publisher of the volume which includes that paper.

The core of Turing's achievement, containing his incompleteness and partial completeness results is contained in §9 of his paper. A more accessible presentation of this is contained in the Technical Appendix to Feferman [1988], but only the summary in the main part of [1988] is reproduced here.

PREFACE (by Solomon Feferman)

3. The purpose of ordinal logics

The purpose of ordinal logics was to try to overcome the incompleteness phenomena discovered by Gödel, by means of transfinite iteration of principles which serve to overcome incompleteness locally. Turing's investigation was characteristically original and penetrating.

3.1. *The rough idea*

With each sufficiently correct effectively generated formal axiomatic system (or "logic") L is associated a true but unprovable statement A_L of the (Π_1^0) form $\forall x R(x)$, expressing that a certain (primitive) recursive property R holds for all integers x . If we start with an initially given system L_1 whose theorems concerning integers are all correct, and adjoin A_{L_1} (call it A_1) to form $L_2 (= L_1 \cup \{A_1\})$, then Gödel's incompleteness result applies again to L_2 , so that associated with L_2 is the true but unprovable A_{L_2} (call it A_2). We can iterate this construction arbitrarily often, to form $L_n = L_1 \cup \{A_1, \dots, A_{n-1}\}$. But $L_\omega \cup \{A_1, \dots, A_n, \dots\}$

is still effectively generated (and correct), so we must proceed further into the transfinite in order to overcome incompleteness. In order to maintain effective generation, one will pass to a transfinite limit ordinal α and system L_α only when α is the limit of an effectively presented sequence $\alpha_1, \alpha_2, \dots, \alpha_n, \dots$ and the systems L_{α_n} are already obtained.

3.2. Making it precise

In order to make this precise, one has to deal with a system of effective representations of ordinals in the integers. Such a system O of *constructive notations a for ordinals α* had been developed by Church and Kleene in 1935 (published in their [1937]) in the framework of the λ -calculus. The idea of Turing's thesis was to investigate the construction and degree of completeness of sequences $\Lambda = \langle L_a \mid a \in O \rangle$ of logics associated with constructive notations a for ordinals, which would increase in strength as the ordinal of a increased, thus overcoming incompleteness – at least locally. The main question was whether one could thereby overcome incompleteness globally. A secondary question was whether such a logic would be *invariant*, i.e. whether the extent of $\Lambda(a)$, the set of theorems of L_a , could depend only on the classical ordinal α associated with a ; this question must be considered since there are generally many notations a for the same α . Turing considered several natural ways in which ordinal logics could be constructed: (i) Λ_P , obtained by successively adjoining statements directly overcoming Gödel incompleteness at each stage a ; (ii) Λ_H , a form of transfinite type theory; and (iii) Λ_G (after Gentzen), obtained by adjoining principles of transfinite induction and recursion up to α at each level $\Lambda_G(a)$.

3.3. The main results

These are contained in §9 of Turing's paper, and are:

- (1) Λ_P is complete for true Π_1^0 statements
- (2) under quite general conditions, an ordinal logic Λ can't be both invariant and complete (even for Π_1^0 statements).

Thus, e.g., Λ_P , being Π_1^0 complete, is not invariant, while Λ_H is necessarily incomplete since it is invariant. Turing had hoped to strengthen (1) to a completeness result for true Π_2^0 sentences, i.e. statements of the form $\forall x \exists y R(x, y)$ with R primitive recursive. This class formally includes various statements of mathematical interest, such as the Riemann Hypothesis (by Turing's analysis of the problem). However, he was unable to achieve such an improved completeness result, and indeed subsequent work of Feferman [1962] showed that Λ_P is incomplete for Π_2^0 sentences. Nevertheless, Turing's (1) could have been interpreted as meeting the initial aim of “overcoming” the incompleteness phenomenon discovered by Gödel, since these only concerned true but unprovable

Π_1^0 statements. Even so, Turing was rightly dissatisfied with this partial completeness result, since it shifted the problem of settling the truth of Π_1^0 statements by axiomatic means to that of recognising whether what appears to be a notation a for a constructive ordinal actually is one; and that problem is at least as complicated as determining which Π_1^0 statements are true. (For, if a is formally given as a limit of a sequence $a(n)$, $n = 1, 2, 3, \dots$, then we have $a \in O$ if and only if $\forall x[a(x) \in O \text{ and } a(x) \text{ represents an increasing sequence}]$, with x ranging over integers. Even the question whether a represents ω is already at least as complicated as the most general Π_1^0 problem. And, as would be shown later by Kleene [1955], the problem, given any a , whether $a \in O$, is more complicated than any arithmetical problem, even using an unlimited number of numerical quantifiers.) Turing was also rightly disappointed with his general incompleteness result (2) for invariant logics. However, he had succeeded in a relatively brief time in making remarkable progress on the topic proposed by Church and had laid the ground for all future investigations in this direction.

4. The shape of things to come

Turing's paper contains several interesting digressions and original observations; there are also some curious failures on his part to observe the obvious. These aspects are reviewed here, the more technical points being recast in modern recursion-theoretic terms.

4.1. It has already been mentioned that Turing shows (§3) the Riemann Hypothesis to be equivalent to a Π_2^0 statement $\forall x \exists y R(x, y)$ with R primitive recursive. This result was quite striking for the time, although it was later improved to Π_1^0 by Kreisel⁷.

Turing further shows that statements of the form $\forall x(F(x) = 0)$ with F general recursive are Π_2^0 . This arises from the fact that each such F can be put in the form $F(x) = U(\min y T(e, x, y))$, where U and T are primitive recursive, arising in the usual way from the fact that e is the Gödel number of a Turing machine M_e that computes F . In fact, a slightly altered argument would have put it into Π_1^0 form (see Feferman [1988], p. 126).

What is more on the mark is Turing's next assertion that each statement “machine M is circle-free”, is Π_2^0 , and conversely, that every Π_2^0 statement is equivalent to one of this form. By definition (in his [1937]), M is *circle-free* if it computes a total function having only the values of 0 and 1 (thus representing a real number in binary form). But by composing any partial function F with the sign function $sg(x + 1) = 1$, $sg(0) = 0$, we see that F is total if and only if $sg(F)$ is

⁷ A simple explicit Π_1^0 representation is given in the paper of Davis, Matijasevic and Robinson [1976], p. 335.

total. Hence the class of statements of the form “ M_e computes a total function” (with no restriction on values), ranges through Π_2^0 . Turing could have moved immediately to this conclusion simply by observing that M_e computes a total function if and only if $\forall x \exists y T(e, x, y)$.

4.2. Next, the brief section §4 contains a striking new idea put to a curious use. The aim here is to produce a problem which is not Π_2^0 . This is trivial by a cardinality argument, but instead, Turing introduces a new notion (which is to change the face of recursion theory) namely, that of computability relative to an *oracle*. He begins by saying: “Let us suppose that we are supplied with some unspecified means of solving number-theoretic [Π_2^0] problems; a kind of oracle as it were.... With the help of the oracle we could form a new kind of machine (call them *o*-machines), having as one of its fundamental processes that of solving a given number-theoretic problem.” He then shows more specifically how to define computability by an *o*-machine and, by a direct extension of his argument in [1937], that the problem of determining “whether an *o*-machine is *o*-circle free” is not solvable by an *o*-machine and hence not by the oracle *o* itself.

Turing did nothing further with the idea of *o*-machines, either in this paper or afterwards. But Post [1944] took it as his basic notion for a theory of degrees of unsolvability and properly credited Turing with the result that for any problem (about integers) there is another of higher degree of unsolvability. Eventually, the idea of transforming computability from an absolute notion to a relative notion would serve to open up the entire subject of generalised recursion theory.

4.3. Later (in §9) after Turing defines ordinal logics $\Lambda = \langle L_a \mid a \in O \rangle$, he takes as one of his main aims that of establishing completeness with respect to Π_2^0 propositions, i.e. of showing that if A is Π_2^0 and true then L_a proves A for some $a \in O$. Now, having already pointed out (§7) that O is not computable (to say the least, as we know), Turing asserts: “We might hope to obtain some intellectually satisfying system of logical inference [for deriving Π_2^0 statements] with some ordinal logic. Gödel’s theorem shows that such a system cannot be wholly mechanical; but with a complete ordinal logic we should be able to confine the nonmechanical steps entirely to verifications that particular formulae [of the λ -calculus] are ordinal formulae.” (Here Turing prefigures his later extended discussion of “the purpose of ordinal logics” which we take up below.) This statement directly follows a brief discussion as to what problems could be solved if we had an oracle for telling us, given a whether or not $a \in O$. But he does not put these together to analyse the logical complexity of $\exists a [a \in O \wedge L_a \vdash A]$ relative to such an oracle *o* (simply that it is Σ_1^0 relative to O).

Immediately after the quoted expression of hope, Turing says: “We might also expect to obtain an interesting classification of number-theoretic [Π_2^0] theorems

according to ‘depth’. A theorem which required an ordinal α to prove it would be deeper than one which could be proved by the use of an ordinal β less than α .” He goes on to say that “however, this presupposes more than is justified”, and carries the idea no further. But here Turing anticipated, at least programmatically, the classification by ordinals of the provably (total) recursive functions of various formal systems, obtained later by proof-theoretical work (see Feferman [1977]). This has been carried over to the classification by depth (or logical strength) of Π_2^0 statements emerging from combinatorial mathematics (see Paris-Harrington [1977] and the survey paper of Simpson [1986]).

4.4. In §10 of Turing’s paper, there is another digression, this time concerning “constructive” analogues of Cantor’s continuum hypothesis, the set-theoretic formulation of which is that there is a 1–1 correspondence between the set $P(\omega)$ of all subsets of ω (or equivalently of all sequences of 0’s and 1’s) and the set of all ordinals less than the least uncountable ordinal ω_1 . Here, for the constructive analogue of $P(\omega)$, Turing takes the set of all computable sequences of 0’s and 1’s (or the description numbers of machines which compute these sequences), and for ω_1 , he substitutes ω_1^{CK} , the least ordinal not constructibly countable in the sense of Church–Kleene, i.e. the least ordinal not represented by a notation in O . Then he asks whether it is possible to set up a computable one-one correspondence between these sets; more precisely, to find a (partial) recursive function F such that for each $a \in O$ and each n , $F(a, n)$ is 0 or 1 and such that $|a| = |a'|$ (i.e. a, a' represent the same ordinal) if and only if $\forall n[F(a, n) = F(a', n)]$. The answer, as Turing shows, is negative; the proof, which is not difficult, transfers a technique that he had applied in §9 to establish the incompleteness of invariant ordinal logics (see the Appendix in Feferman [1988] for more details).

As Turing points out, there is “... great ambiguity concerning what the constructive analogue of the continuum hypothesis should be”, and he addresses only one possible formulation of it. He says that the suggestion for this came indirectly from F. Bernstein and that a related problem was suggested by Bernays. But Turing might also have been inspired by Hilbert’s 1926 paper, “On the Infinite” (to which he refers in a different connection), where Hilbert attempted to establish an ordinal-recursive classification of integer functions in his abortive “solution” of the continuum problem.

In any case, here again Turing anticipates later work, on the classification of recursive functions by means of hierarchies. It is customary in that work to consider some simple relative computability relation $f \leq g$ between functions, such as that f is primitive recursive in g (or even weaker), and to seek assignments $f_a = \lambda n F(a, n)$ to each $a \in O$ with the property that f_a increases with the ordinal of a . Here the invariance required by Turing for his version of the continuum hypothesis is weakened to $|a| = |a'| \Rightarrow f_a \equiv f_{a'}$ (i.e. $f_a \leq f_{a'} \leq f_a$). It turns

out that the general theory of such classifications, with both incompleteness and completeness results, runs entirely parallel to the theory of ordinal logics (see Feferman [1962a] and the Technical Appendix of [1988]).

5. The significance of Turing's paper

It is best to begin by quoting from Turing's own two page discussion (§11): "Mathematical reasoning may be regarded rather schematically as the exercise of a combination of two faculties, which we may call *intuition* and *ingenuity*." (There is a nice footnote to this sentence: "We are leaving out of account the most important faculty which distinguishes topics of interest from others...") He goes on to explain that "intuition consists in making spontaneous judgements which are not the result of conscious trains of reasoning. These judgements are often but by no means invariably correct... The exercise of ingenuity in mathematics consists in aiding the intuition through suitable arrangements of propositions. ... When these are really well arranged the validity of the intuitive steps which are required cannot seriously be doubted." In Turing's view, there is no sharp, objective line between these two "faculties" – the parts they play differ "from occasion to occasion and mathematician to mathematician". Formal logic helps remove this "arbitrariness"; the formal rules are supposed to be chosen so that inferences are always intuitively valid. Moreover, the exercise of ingenuity is then given more shape in the search for admissible chains of inference to make up a proof. "In pre-Gödel times it was thought by some that it would... be possible to carry this programme to such a point that... the necessity for intuition would then be entirely eliminated." But Gödel's incompleteness theorems have shown this is impossible, and one turns naturally instead to "nonconstructive" systems of logic in which "not all the steps in a proof are mechanical, some being intuitive." Ordinal logics provide examples of such: "When we have an ordinal logic, we are in a position to prove number-theoretic theorems by the intuitive steps of recognising formulae [of the λ -calculus] as ordinal formulae [representing well-orderings]. ... We want it to show quite clearly when a step makes use of intuition and when it is purely formal. The strain put on the intuition should be minimal. Most important of all, it must be beyond all reasonable doubt that the logic leads to correct results whenever the intuitive steps [i.e. recognition of notations for ordinals] are correct." Turing concludes the discussion by considering his ordinal logics Λ_P and Λ_H with respect to this last criterion, and after putting them in question, moves on at the end of his paper to consider a new type of ordinal logic Λ_G . His judgments about these particular logics do not concern us here; rather we shall examine his general conception about the whole enterprise.

In modern logical discussions, ingenuity in its normal sense is put outside the purview of the subject, just as Turing set the question of the interest of

results aside, though for working mathematicians both of these are critical to what counts as “good” mathematics. Logic simply hopes to answer the question: “What counts as mathematics?”, while disregarding all questions of value. In “pre-Gödel times”, it thought to find an answer in that which can be formalised, or, more fully, that which can be formalised in an axiom system which is justified on the grounds of some basic mathematical conception. However, even in those times, the question of *which* formal systems were so justified received no common answer. And even if it had turned out that one could find a complete formal system for mathematics, the question of justification would still remain – though no doubt it would have seemed less compelling. In any case, Gödel’s incompleteness theorem brought to centre stage the question of what leads one to accept a formal system for the (necessarily partial) representation of mathematics. It is here that “intuition” would have to play its role, both in judging the acceptability of any proposed systems and in searching for new such systems. However, there was one aspect of the incompleteness theorem that suggested how the day might be saved without having to over-exercise the intuition: Gödel’s examples of formally undecidable propositions can be decided directly by informal considerations. If a statement φ which “says” of itself that it is underivable in L is shown to be underivable in L , then it is evidently true and ought to be added to L ; it is just that somehow such were “overlooked”, so that the expanded system ought to be deemed acceptable if L is. Thus, if the passage from L to L' is obtained simply by adjoining such evidently correct statements to L , the acceptability of L' follows directly from that of L . Also, if each of a sequence $L_{a_1} \subset L_{a_2} \subset \dots \subset L_{a_n} \subset \dots$ of increasing systems is recognised to be acceptable then $\bigcup_n L_{a_n}$ is evidently acceptable too. Hence, if we start with a basic acceptable system L_1 , it seems that *all* we have to do to overcome Gödel’s incompleteness is to iterate the passage from L to L' transfinitely, as in ordinal logics, to obtain the collection of systems $\langle L_a \mid a \in O \rangle$. But then the whole question of which formal systems ought to be admitted under this process shifts to the question: for which representations a of ordinals is it justified to accept L_a ? Here there is an implicit use of the principle of transfinite induction on the set $\{b \mid b \leq O a\}$ of notations up to and including a , applied to the informal predicate “ L_b is acceptable”, i.e. “ L_b is correct according to a basic mathematical conception”.

Thus the demand on “intuition” in recognising “which formulae are ordinal formulae” is somewhat greater than Turing suggests. But even in his own terms, there is a failure to test his analysis of purpose against reality. Is it a “spontaneous judgement” without *any* “conscious train of reasoning” that leads one to recognise a complicated though computable ordering as being a well-ordering? Can one truly say that of familiar orderings for ε_0 or even ω^ω , let alone the more complicated orderings that have naturally emerged in modern proof theory $(\Gamma_0, \phi_{\varepsilon_\Omega+1}(0), \dots)$? Surely not. And once *some* form of reasoning has to be

admitted, then the whole question of which notations for ordinals are to be accepted (if one is to work with ordinal logics at all) must be re-examined. This was subsequently done, at the suggestion of Kreisel [1958], by restricting attention, successively, to those notations a for which one has a *proof* in L_b for some $b <_O a$ that $a \in O$ (i.e. that a represents a well-ordering). These have come to be called *autonomous ordinal notations*, and the notion of ordinal logic restricted in this way, *autonomous recursive progressions of axiomatic theories*. Kreisel originally applied the notion of autonomous ordinal logics in his analysis of finitism, and I took it up (see Feferman [1964] and [1967]) in a corresponding analysis of predicativity. The idea of an autonomous progression more nearly approximates the process of finding out what is *implicit* in accepting a basic system L_1 , i.e. what one ought to accept, on the same fundamental grounds, if one accepts L_1 . This theme is developed further in Kreisel [1970] and Feferman [1991]. The latter paper introduced a new notion, that of *reflective closure* of an axiomatic theory, which is a more realistic way to explain the idea of what is implicit in accepting a given axiomatic system. A still more realistic way of explaining this concept has been advanced more recently via the concept of the *unfolding* of such a system in Feferman [1996].

It is easily shown that $\bigcup L_a$ [a autonomous] is recursively axiomatizable and hence incomplete by Gödel's theorem. Thus, whatever the value of the notion of autonomous progression (or reflective closure) for describing all that we ought to accept once we have made a basic conceptual commitment, we cannot hope thereby to answer the general question as to which axiomatic theories ought to be accepted according to the explanation, in a logical framework, of what constitutes mathematics. Ordinal logics provided the first model for attacking this question in any systematic way, and perhaps their greatest value lay in demonstrating the possibility of carrying out such an investigation at all. Turing's disappointment in their usefulness, and our latter-day disappointment for more sophisticated reasons, ought not to diminish our appreciation of what, after all, was this remarkable feature of Turing's achievement.

6. Ordinal logics and mechanistic thought

It is natural to move on to relate Turing's work on ordinal logics to his post-war work on machine intelligence and human mental activity (particularly in relation to mathematics). The relevant post-war work is Turing's NPL report [1948] and the further sources provided by Hodges [1983] (especially Chapter 6). 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 behaviour. His idea seemed to be: if intelligent human activity depends only on the structure of the brain as a network of cells that are either activated or unactivated (and not on

its physical embodiment in tissue) then whatever humans can do in this respect can, in principle, be mimicked by machines. But even for Turing, the structure of the brain qua machine, as complicated as it may be, does not by itself suffice for intelligent behaviour. For that, he says, one also needs external instruction and internal initiative. Similarly, machines must be taught to think, be rewarded for success and punished for failure, so that they may learn from experience. Here one has a division between what is provided by the mechanism *per se* and what must be brought from outside in the form of a program that instructs the machine what to do. This echoes the division in ordinal logics between what is provided by the formal application of axioms and rules in any logic L_a and what must come from outside in order to recognise (by “intuition” or an “oracle”) a as an ordinal notation that unlocks the L_a machinery. Also, once inside L_a , the application of “ingenuity” in searching for a proof of a conjecture may be compared with the required “initiative” for intelligent behaviour. However, I would not push this analogy too far, since it is already a bit strained.

In sum, one might regard Turing’s work on ordinal logics as a temporary shunting off from his main track of thought, from 1936–37 on through his war work on mechanical cryptanalysis and then, after the war, with the design of computers and the analysis of intelligent behaviour in mechanical terms. Turing never tried to develop an over-all philosophy of mathematics and in the end did not seem to be really troubled by the problems that Gödel’s theorem raised for a mechanistic theory of mind. Indeed, I suspect from the history that he did not really have his heart in the Ph.D. work under Church, though the idea of Church’s suggestion certainly appealed to him and, once engaged on it, he gave it the fullest of his mind.

An interesting alternative point of view has recently been put forward by Hodges [1997]. He writes “I fell into this assumption in *Alan Turing: the Enigma*, [1983] essentially because I followed Turing’s own later standpoint. But I now consider that *at the time*, Turing saw himself steaming straight ahead with the analysis of the mind, by studying a question complementary to ‘On computable numbers...’. Turing asked in this paper whether it is possible to formalise those actions of the mind which are *not* those of following a definite method: mental actions one might call creative or original in nature. In particular, Turing focussed on the action of seeing the truth of one of Gödel’s unprovable assertions.”

This Page Intentionally Left Blank

SYSTEMS OF LOGIC BASED ON ORDINALS†

By A. M. TURING.

[Received 31 May, 1938.—Read 16 June, 1938.]

Introduction	161
1. The conversion calculus. Gödel representations	162
2. Effective calculability. Abbreviation of treatment	166
3. Number-theoretic theorems	168
4. A type of problem which is not number-theoretic	172
5. Syntactical theorems as number-theoretic theorems	174
6. Logic formulae	174
7. Ordinals	178
8. Ordinal logics	189
9. Completeness questions	198
10. The continuum hypothesis. A digression	213
11. The purpose of ordinal logics	214
12. Gentzen type ordinal logics	217
Index of definitions	225
Bibliography	228

The well-known theorem of Gödel (Gödel [1], [2]) shows that every system of logic is in a certain sense incomplete, but at the same time it indicates means whereby from a system L of logic a more complete system L' may be obtained. By repeating the process we get a sequence $L, L_1 = L', L_2 = L_1', \dots$ each more complete than the preceding. A logic L_ω may then be constructed in which the provable theorems are the totality of theorems provable with the help of the logics L, L_1, L_2, \dots . We may then form $L_{2\omega}$ related to L_ω in the same way as L_ω was related to L . Proceeding in this way we can associate a system of logic with any constructive ordinal‡. It may be asked whether a sequence of logics of this kind is complete in the sense that to any problem A there corresponds

† This paper represents work done while a Jane Eliza Procter Visiting Fellow at Princeton University, where the author received most valuable advice and assistance from Prof. Alonzo Church.

‡ The situation is not quite so simple as is suggested by this crude argument. See pages 189–193, 202, 203.

an ordinal α such that A is solvable by means of the logic L_α . I propose to investigate this question in a rather more general case, and to give some other examples of ways in which systems of logic may be associated with constructive ordinals.

1. *The calculus of conversion. Gödel representations.*

It will be convenient to be able to use the “conversion calculus” of Church for the description of functions and for some other purposes. This will make greater clarity and simplicity of expression possible. I give a short account of this calculus. For detailed descriptions see Church [3], [2], Kleene [1], Church and Rosser [1].

The formulae of the calculus are formed from the symbols $\{$, $\}$, $($, $)$, $[$, $]$, λ , δ , and an infinite list of others called variables; we shall take for our infinite list $a, b, \dots, z, x', x'', \dots$. Certain finite sequences of such symbols are called *well-formed formulae* (abbreviated to W.F.F.); we define this class inductively, and define simultaneously the free and the bound variables of a W.F.F. Any variable is a W.F.F.; it is its only free variable, and it has no bound variables. δ is a W.F.F. and has no free or bound variables. If \mathbf{M} and \mathbf{N} are W.F.F. then $\{\mathbf{M}\}(\mathbf{N})$ is a W.F.F., whose free variables are the free variables of \mathbf{M} together with the free variables of \mathbf{N} , and whose bound variables are the bound variables of \mathbf{M} together with those of \mathbf{N} . If \mathbf{M} is a W.F.F. and \mathbf{V} is one of its free variables, then $\lambda\mathbf{V}[\mathbf{M}]$ is a W.F.F. whose free variables are those of \mathbf{M} with the exception of \mathbf{V} , and whose bound variables are those of \mathbf{M} together with \mathbf{V} . No sequence of symbols is a W.F.F. except in consequence of these three statements.

In metamathematical statements we use heavy type letters to stand for variable or undetermined formulae, as was done in the last paragraph, and in future such letters will stand for well-formed formulae unless otherwise stated. Small letters in heavy type will stand for formulae representing undetermined positive integers (see below).

A W.F.F. is said to be in normal form if it has no parts of the form $\{\lambda\mathbf{V}[\mathbf{M}]\}(\mathbf{N})$ and none of the form $\{\{\delta\}(\mathbf{M})\}(\mathbf{N})$, where \mathbf{M} and \mathbf{N} have no free variables.

We say that one W.F.F. is *immediately convertible* into another if it is obtained from it either by:

- (i) Replacing one occurrence of a well-formed part $\lambda\mathbf{V}[\mathbf{M}]$ by $\lambda\mathbf{U}[\mathbf{N}]$, where the variable \mathbf{U} does not occur in \mathbf{M} , and \mathbf{N} is obtained from \mathbf{M} by replacing the variable \mathbf{V} by \mathbf{U} throughout.

(ii) Replacing a well-formed part $\{\lambda \mathbf{V}[\mathbf{M}]\}(\mathbf{N})$ by the formula which is obtained from \mathbf{M} by replacing \mathbf{V} by \mathbf{N} throughout, provided that the bound variables of \mathbf{M} are distinct both from \mathbf{V} and from the free variables of \mathbf{N} .

(iii) The process inverse to (ii).

(iv) Replacing a well-formed part $\{\{\delta\}(\mathbf{M})\}(\mathbf{M})$ by

$$\lambda f \left[\lambda x \left[\{f\} (\{f\}(x)) \right] \right]$$

if \mathbf{M} is in normal form and has no free variables.

(v) Replacing a well-formed part $\{\{\delta\}(\mathbf{M})\}(\mathbf{N})$ by

$$\lambda f \left[\lambda x [\{f\}(x)] \right]$$

if \mathbf{M} and \mathbf{N} are in normal form, are not transformable into one another by repeated application of (i), and have no free variables.

(vi) The process inverse to (iv).

(vii) The process inverse to (v).

These rules could have been expressed in such a way that in no case could there be any doubt about the admissibility or the result of the transformation [in particular this can be done in the case of process (v)].

A formula \mathbf{A} is said to be *convertible* into another \mathbf{B} (abbreviated to “ \mathbf{A} conv \mathbf{B} ”) if there is a finite chain of immediate conversions leading from one formula to the other. It is easily seen that the relation of convertibility is an equivalence relation, *i.e.* it is symmetric, transitive, and reflexive.

Since the formulae are liable to be very lengthy, we need means for abbreviating them. If we wish to introduce a particular letter as an abbreviation for a particular lengthy formula we write the letter followed by “ \rightarrow ” and then by the formula, thus

$$I \rightarrow \lambda x[x]$$

indicates that I is an abbreviation for $\lambda x[x]$. We also use the arrow in less sharply defined senses, but never so as to cause any real confusion. In these cases the meaning of the arrow may be rendered by the words “stands for”.

If a formula \mathbf{F} is, or is represented by, a single symbol we abbreviate $\{\mathbf{F}\}(\mathbf{X})$ to $\mathbf{F}(\mathbf{X})$. A formula $\{\{\mathbf{F}\}(\mathbf{X})\}(\mathbf{Y})$ may be abbreviated to

$$\{\mathbf{F}\}(\mathbf{X}, \mathbf{Y}),$$

or to $\mathbf{F}(\mathbf{X}, \mathbf{Y})$ if \mathbf{F} is, or is represented by, a single symbol. Similarly for $\{\{\{\mathbf{F}\}(\mathbf{X})\}(\mathbf{Y})\}(\mathbf{Z})$, etc. A formula $\lambda \mathbf{V}_1 [\lambda \mathbf{V}_2 \dots [\lambda \mathbf{V}_r [\mathbf{M}] \dots]]$ may be abbreviated to $\lambda \mathbf{V}_1 \mathbf{V}_2 \dots \mathbf{V}_r, \mathbf{M}$.

We have not as yet assigned any meanings to our formulae, and we do not intend to do so in general. An exception may be made for the case of the positive integers, which are very conveniently represented by the formulae $\lambda f x. f(x)$, $\lambda f x. f(f(x))$, In fact we introduce the abbreviations

$$1 \rightarrow \lambda f x. f(x)$$

$$2 \rightarrow \lambda f x. f(f(x))$$

$$3 \rightarrow \lambda f x. f(f(f(x))), \text{ etc.,}$$

and we also say, for example, that $\lambda f x. f(f(x))$, or in full

$$\lambda f [\lambda x [\{f\}(\{f\}(x))]],$$

represents the positive integer 2. Later we shall allow certain formulae to represent ordinals, but otherwise we leave them without explicit meaning; an implicit meaning may be suggested by the abbreviations used. In any case where any meaning is assigned to formulae it is desirable that the meaning should be invariant under conversion. Our definitions of the positive integers do not violate this requirement, since it may be proved that no two formulae representing different positive integers are convertible the one into the other.

In connection with the positive integers we introduce the abbreviation

$$S \rightarrow \lambda u f x. f(u(f, x)).$$

This formula has the property that, if \mathbf{n} represents a positive integer, $S(\mathbf{n})$ is convertible to a formula representing its successor†.

Formulae representing undetermined positive integers will be represented by small letters in heavy type, and we adopt once for all the

† This follows from (A) below.

convention that, if a small letter, n say, stands for a positive integer, then the same letter in heavy type, \mathbf{n} , stands for the formula representing the positive integer. When no confusion arises from so doing, we shall not trouble to distinguish between an integer and the formula which represents it.

Suppose that $f(n)$ is a function of positive integers taking positive integers as values, and that there is a W.F.F. \mathbf{F} not containing δ such that, for each positive integer n , $\mathbf{F}(\mathbf{n})$ is convertible to the formula representing $f(n)$. We shall then say that $f(n)$ is *λ -definable* or *formally definable*, and that \mathbf{F} *formally defines* $f(n)$. Similar conventions are used for functions of more than one variable. The sum function is, for instance, formally defined by $\lambda abfx.a(f, b(f, x))$; in fact, for any positive integers m, n, p for which $m+n=p$, we have

$$\{\lambda abfx.a(f, b(f, x))\}(\mathbf{m}, \mathbf{n}) \text{ conv } \mathbf{p}.$$

In order to emphasize this relation we introduce the abbreviation

$$\mathbf{X} + \mathbf{Y} \rightarrow \{\lambda abfx.a(f, b(f, x))\}(\mathbf{X}, \mathbf{Y})$$

and we shall use similar notations for sums of three or more terms, products, etc.

For any W.F.F. \mathbf{G} we shall say that \mathbf{G} *enumerates* the sequence $\mathbf{G}(1), \mathbf{G}(2), \dots$ and any other sequence whose terms are convertible to those of this sequence.

When a formula is convertible to another which is in normal form, the second is described as a *normal form* of the first, which is then said to *have a normal form*. I quote here some of the more important theorems concerning normal forms.

(A) *If a formula has two normal forms they are convertible into one another by the use of (i) alone.* (Church and Rosser [1], 479, 481.)

(B) *If a formula has a normal form then every well-formed part of it has a normal form.* (Church and Rosser [1], 480–481.)

(C) *There is (demonstrably) no process whereby it can be said of a formula whether it has a normal form.* (Church [3], 360, Theorem XVIII.)

We often need to be able to describe formulae by means of positive integers. The method used here is due to Gödel (Gödel [1]). To each single symbol s of the calculus we assign an integer $r[s]$ as in the table below.

s	{, (, or [,],), or]}	λ	δ	a	...	z	x'	x''	x'''	...	
$r[s]$	1	2	3	4	5	...	30	31	32	33	...

If s_1, s_2, \dots, s_k is a sequence of symbols, then $2^{r[s_1]} 3^{r[s_2]} \dots p_k^{r[s_k]}$ (where p_k is the k -th prime number) is called the *Gödel representation* (G.R.) of that sequence of symbols. No two W.F.F. have the same G.R.

Two theorems on G.R. of W.F.F. are quoted here.

(D) *There is a W.F.F. "form" such that if a is the G.R. of a W.F.F. \mathbf{A} without free variables, then form (a) conv \mathbf{A} .* (This follows from a similar theorem to be found in Church [3], 53–66. Metads are used there in place of G.R.)

(E) *There is a W.F.F. Gr such that, if \mathbf{A} is a W.F.F. with a normal form without free variables, then $\text{Gr}(\mathbf{A}) \text{ conv } a$, where a is the G.R. of a normal form of \mathbf{A} .* [Church [3], 53, 66, as (D).]

2. Effective calculability. Abbreviation of treatment.

A function is said to be “effectively calculable” if its values can be found by some purely mechanical process. Although it is fairly easy to get an intuitive grasp of this idea, it is nevertheless desirable to have some more definite, mathematically expressible definition. Such a definition was first given by Gödel at Princeton in 1934 (Gödel [2], 26), following in part an unpublished suggestion of Herbrand, and has since been developed by Kleene [2]). These functions were described as “general recursive” by Gödel. We shall not be much concerned here with this particular definition. Another definition of effective calculability has been given by Church (Church [3], 356–358), who identifies it with λ -definability. The author has recently suggested a definition corresponding more closely to the intuitive idea (Turing [1], see also Post [1]). It was stated above that “a function is effectively calculable if its values can be found by some purely mechanical process”. We may take this statement literally, understanding by a purely mechanical process one which could be carried out by a machine. It is possible to give a mathematical description, in a certain normal form, of the structures of these machines. The development of these ideas leads to the author’s definition of a computable function, and to an identification of computability† with effective calculability. It is not difficult, though somewhat laborious, to prove that these three definitions are equivalent (Kleene [3], Turing [2]).

† We shall use the expression “computable function” to mean a function calculable by a machine, and we let “effectively calculable” refer to the intuitive idea without particular identification with any one of these definitions. We do not restrict the values taken by a computable function to be natural numbers; we may for instance have computable propositional functions.

In the present paper we shall make considerable use of Church's identification of effective calculability with λ -definability, or, what comes to the same thing, of the identification with computability and one of the equivalence theorems. In most cases where we have to deal with an effectively calculable function, we shall introduce the corresponding W.F.F. with some such phrase as "the function f is effectively calculable, let F be a formula λ defining it", or "let F be a formula such that $F(\mathbf{n})$ is convertible to . . . whenever \mathbf{n} represents a positive integer". In such cases there is no difficulty in seeing how a machine could in principle be designed to calculate the values of the function concerned; and, assuming this done, the equivalence theorem can be applied. A statement of what the formula F actually is may be omitted. We may immediately introduce on this basis a W.F.F. ϖ with the property that

$$\varpi(\mathbf{m}, \mathbf{n}) \text{ conv } \mathbf{r},$$

if r is the greatest positive integer, if any, for which m^r divides n and r is 1 if there is none. We also introduce Dt with the properties

$$\text{Dt}(\mathbf{n}, \mathbf{n}) \text{ conv } 3,$$

$$\text{Dt}(\mathbf{n} + \mathbf{m}, \mathbf{n}) \text{ conv } 2,$$

$$\text{Dt}(\mathbf{n}, \mathbf{n} + \mathbf{m}) \text{ conv } 1.$$

There is another point to be made clear in connection with the point of view that we are adopting. It is intended that all proofs that are given should be regarded no more critically than proofs in classical analysis. The subject matter, roughly speaking, is constructive systems of logic, but since the purpose is directed towards choosing a particular constructive system of logic for practical use, an attempt at this stage to put our theorems into constructive form would be putting the cart before the horse.

Those computable functions which take only the values 0 and 1 are of particular importance, since they determine and are determined by computable properties, as may be seen by replacing "0" and "1" by "true" and "false". But, besides this type of property, we may have to consider a different type, which is, roughly speaking, less constructive than the computable properties, but more so than the general predicates of classical mathematics. Suppose that we have a computable function of the natural numbers taking natural numbers as values, then corresponding to this function there is the property of being a value of the function. Such a property we shall describe as "axiomatic"; the reason for using this term is that it is possible to define such a property by giving a set of axioms, the property to hold for a given argument if and only if it is possible to deduce that it holds from the axioms.

Axiomatic properties may also be characterized in this way. A property ψ of positive integers is axiomatic if and only if there is a computable property ϕ of two positive integers, such that $\psi(x)$ is true if and only if there is a positive integer y such that $\phi(x, y)$ is true. Or again ψ is axiomatic if and only if there is a W.F.F. \mathbf{F} such that $\psi(n)$ is true if and only if $\mathbf{F}(n)$ conv 2.

3. Number-theoretic theorems.

By a *number-theoretic theorem*† we shall mean a theorem of the form “ $\theta(x)$ vanishes for infinitely many natural numbers x ”, where $\theta(x)$ is a primitive recursive‡ function.

We shall say that a problem is number-theoretic if it has been shown that any solution of the problem may be put in the form of a proof of one or more number-theoretic theorems. More accurately we may say that a class of problems is number-theoretic if the solution of any one of them can be transformed (by a uniform process) into the form of proofs of number-theoretic theorems.

I shall now draw a few consequences from the definition of “number theoretic theorems”, and in section 5 I shall try to justify confining our consideration to this type of problem.

† I believe that there is no generally accepted meaning for this term, but it should be noticed that we are using it in a rather restricted sense. The most generally accepted meaning is probably this: suppose that we take an arbitrary formula of the functional calculus of the first order and replace the function variables by primitive recursive relations. The resulting formula represents a typical number-theoretic theorem in this (more general) sense.

‡ Primitive recursive functions of natural numbers are defined inductively as follows. Suppose that $f(x_1, \dots, x_{n-1})$, $g(x_1, \dots, x_n)$, $h(x_1, \dots, x_{n+1})$ are primitive recursive, then $\phi(x_1, \dots, x_n)$ is primitive recursive if it is defined by one of the sets of equations (a) to (e).

- (a) $\phi(x_1, \dots, x_n) = h(x_1, \dots, x_{m-1}, g(x_1, \dots, x_n), x_{m+1}, \dots, x_{n-1}, x_m)$ ($1 \leq m \leq n$);
- (b) $\phi(x_1, \dots, x_n) = f(x_2, \dots, x_n);$
- (c) $\phi(x_1) = a$, where $n = 1$ and a is some particular natural number;
- (d) $\phi(x_1) = x_1 + 1$ ($n = 1$);
- (e) $\phi(x_1, \dots, x_{n-1}, 0) = f(x_1, \dots, x_{n-1});$
 $\phi(x_1, \dots, x_{n-1}, x_n + 1) = h(x_1, \dots, x_n, \phi(x_1, \dots, x_n)).$

The class of primitive recursive functions is more restricted than the class of computable functions, but it has the advantage that there is a process whereby it can be said of a set of equations whether it defines a primitive recursive function in the manner described above.

An alternative form for number-theoretic theorems is “for each natural number x there exists a natural number y such that $\phi(x, y)$ vanishes”, where $\phi(x, y)$ is primitive recursive. In other words, there is a rule whereby, given the function $\theta(x)$, we can find a function $\phi(x, y)$, or given $\phi(x, y)$, we can find a function $\theta(x)$, such that “ $\theta(x)$ vanishes infinitely often” is a necessary and sufficient condition for “for each x there is a y such that $\phi(x, y) = 0$ ”. In fact, given $\theta(x)$, we define

$$\phi(x, y) = \theta(x) + a(x, y),$$

where $a(x, y)$ is the (primitive recursive) function with the properties

$$\begin{aligned} a(x, y) &= 1 \quad (y \leq x), \\ &= 0 \quad (y > x). \end{aligned}$$

If on the other hand we are given $\phi(x, y)$ we define $\theta(x)$ by the equations

$$\theta_1(0) = 3,$$

$$\theta_1(x+1) = 2^{(1+\varpi_2(\theta_1(x)))} \sigma(\phi(\varpi_3(\theta_1(x))-1, \varpi_2(\theta_1(x)))) 3^{\varpi_3(\theta_1(x))+1-\sigma(\phi(\varpi_3(\theta_1(x))-1, \varpi_2(\theta_1(x))))},$$

$$\theta(x) = \phi(\varpi_3(\theta_1(x))-1, \varpi_2(\theta_1(x))),$$

where $\varpi_r(x)$ is defined so as to mean “the largest s for which r^s divides x ”. The function $\sigma(x)$ is defined by the equations $\sigma(0) = 0$, $\sigma(x+1) = 1$. It is easily verified that the functions so defined have the desired properties.

We shall now show that questions about the truth of the statements of the form “does $f(x)$ vanish identically”, where $f(x)$ is a computable function, can be reduced to questions about the truth of number-theoretic theorems. It is understood that in each case the rule for the calculation of $f(x)$ is given and that we are satisfied that this rule is valid, *i.e.* that the machine which should calculate $f(x)$ is circle free (Turing [1], 233). The function $f(x)$, being computable, is general recursive in the Herbrand-Gödel sense, and therefore, by a general theorem due to Kleene†, is expressible in the form

$$\psi(\epsilon y[\phi(x, y) = 0]), \quad (3.2)$$

where $\epsilon y[\mathfrak{U}(y)]$ means “the least y for which $\mathfrak{U}(y)$ is true” and $\psi(y)$ and $\phi(x, y)$ are primitive recursive functions. Without loss of generality, we may suppose that the functions ϕ, ψ take only the values 0, 1. Then, if

† Kleene [3], 727. This result is really superfluous for our purpose, since the proof that every computable function is general recursive proceeds by showing that these functions are of the form (3.2). (Turing [2], 161).

we define $\rho(x)$ by the equations (3.1) and

$$\rho(0) = \psi(0)(1 - \theta(0)),$$

$$\rho(x+1) = 1 - (1 - \rho(x)) \sigma [1 + \theta(x) - \psi \{ \varpi_2(\theta_1(x)) \}]$$

it will be seen that $f(x)$ vanishes identically if and only if $\rho(x)$ vanishes for infinitely many values of x .

The converse of this result is not quite true. We cannot say that the question about the truth of any number-theoretic theorem is reducible to a question about whether a corresponding computable function vanishes identically; we should have rather to say that it is reducible to the problem of whether a certain machine is circle free and calculates an identically vanishing function. But more is true: every number-theoretic theorem is equivalent to the statement that a corresponding machine is circle free. The behaviour of the machine may be described roughly as follows: the machine is one for the calculation of the primitive recursive function $\theta(x)$ of the number-theoretic problem, except that the results of the calculation are first arranged in a form in which the figures 0 and 1 do not occur, and the machine is then modified so that, whenever it has been found that the function vanishes for some value of the argument, then 0 is printed. The machine is circle free if and only if an infinity of these figures are printed, i.e. if and only if $\theta(x)$ vanishes for infinitely many values of the argument. That, on the other hand, questions of circle freedom may be reduced to questions of the truth of number-theoretic theorems follows from the fact that $\theta(x)$ is primitive recursive when it is defined to have the value 0 if a certain machine \mathcal{M} prints 0 or 1 in its $(x+1)$ -th complete configuration, and to have the value 1 otherwise.

The conversion calculus provides another normal form for the number-theoretic theorems, and the one which we shall find the most convenient to use. Every number-theoretic theorem is equivalent to a statement of the form “ $\mathbf{A}(n)$ is convertible to 2 for every W.F.F. n representing a positive integer”, \mathbf{A} being a W.F.F. determined by the theorem; the property of \mathbf{A} here asserted will be described briefly as “ \mathbf{A} is dual”. Conversely such statements are reducible to number theoretic theorems. The first half of this assertion follows from our results for computable functions, or directly in this way. Since $\theta(x-1)+2$ is primitive recursive, it is formally definable, say, by means of a formula \mathbf{G} . Now there is (Kleene [1], 232) a W.F.F. \mathcal{P} with the property that, if $\mathbf{T}(r)$ is convertible to a formula representing a positive integer for each positive integer r , then $\mathcal{P}(\mathbf{T}, n)$ is convertible to s , where s is the n -th positive integer t (if there is one) for which

$\mathbf{T}(\mathbf{t})$ conv 2; if $\mathbf{T}(\mathbf{t})$ conv 2 for less than n values of t then $\mathcal{P}(\mathbf{T}, \mathbf{n})$ has no normal form. The formula $\mathbf{G}(\mathcal{P}(\mathbf{G}, \mathbf{n}))$ is therefore convertible to 2 if and only if $\theta(x)$ vanishes for at least n values of x , and is convertible to 2 for every positive integer x if and only if $\theta(x)$ vanishes infinitely often. To prove the second half of the assertion, we take Gödel representations for the formulae of the conversion calculus. Let $c(x)$ be 0 if x is the G.R. of 2 (*i.e.* if x is $2^3 \cdot 3^{10} \cdot 5 \cdot 7^3 \cdot 11^{28} \cdot 13 \cdot 17 \cdot 19^{10} \cdot 23^2 \cdot 29 \cdot 31 \cdot 37^{10} \cdot 41^2 \cdot 43 \cdot 47^{28} \cdot 53^2 \cdot 59^2 \cdot 61^2 \cdot 67^2$) and let $c(x)$ be 1 otherwise. Take an enumeration of the G.R. of the formulae into which $\mathbf{A}(\mathbf{m})$ is convertible: let $a(m, n)$ be the n -th number in the enumeration. We can arrange the enumeration so that $a(m, n)$ is primitive recursive. Now the statement that $\mathbf{A}(\mathbf{m})$ is convertible to 2 for every positive integer m is equivalent to the statement that, corresponding to each positive integer m , there is a positive integer n such that $c(a(m, n)) = 0$; and this is number-theoretic.

It is easy to show that a number of unsolved problems, such as the problem of the truth of Fermat's last theorem, are number-theoretic. There are, however, also problems of analysis which are number-theoretic. The Riemann hypothesis gives us an example of this. We denote by $\zeta(s)$ the function defined for $\Re s = \sigma > 1$ by the series $\sum_{n=1}^{\infty} n^{-s}$ and over the rest of the complex plane with the exception of the point $s = 1$ by analytic continuation. The Riemann hypothesis asserts that this function does not vanish in the domain $\sigma > \frac{1}{2}$. It is easily shown that this is equivalent to saying that it does not vanish for $2 > \sigma > \frac{1}{2}$, $\Im s = t > 2$, *i.e.* that it does not vanish inside any rectangle $2 > \sigma > \frac{1}{2} + 1/T$, $T > t > 2$, where T is an integer greater than 2. Now the function satisfies the inequalities

$$\left. \begin{aligned} \left| \zeta(s) - \sum_1^N n^{-s} - \frac{N^{1-s}}{s-1} \right| &< 2t(N-2)^{-\frac{1}{2}}, \quad 2 < \sigma < \frac{1}{2}, \quad t \geq 2, \\ |\zeta(s) - \zeta(s')| &< 60t|s - s'|, \quad 2 < \sigma' < \frac{1}{2}, \quad t' \geq 2, \end{aligned} \right\}$$

and we can define a primitive recursive function $\xi(l, l', m, m', N, M)$ such that

$$\left| \xi(l, l', m, m', N, M) - M \left| \sum_1^N n^{-s} + \frac{N^{1-s}}{s-1} \right| \right| < 2, \quad \left(s = \frac{l}{l'} + i \frac{m}{m'} \right),$$

and therefore, if we put

$$\xi(l, M, m, M, M^2 + 2, M) = X(l, m, M),$$

we have

$$\left| \zeta \left(\frac{l+\vartheta}{M} + i \frac{m+\vartheta}{M} \right) \right| \geq \frac{X(l, m, M) - 122T}{M},$$

provided that

$$\frac{1}{2} + \frac{1}{T} \leq \frac{l-1}{M} < \frac{l+1}{M} < 2 - \frac{1}{M}, \quad 2 < \frac{m-1}{M} < \frac{m+1}{M} < T$$

$$(-1 < \vartheta < 1, \quad -1 < \vartheta' < 1).$$

If we define $B(M, T)$ to be the smallest value of $X(l, m, M)$ for which

$$\frac{1}{2} + \frac{1}{T} + \frac{1}{M} \leq \frac{l}{M} < 2 - \frac{1}{M}, \quad 2 + \frac{1}{M} < \frac{m}{M} < T - \frac{1}{M},$$

then the Riemann hypothesis is true if for each T there is an M satisfying

$$B(M, T) > 122T.$$

If on the other hand there is a T such that, for all M , $B(M, T) \leq 122T$, the Riemann hypothesis is false; for let l_M, m_M be such that

$$X(l_M, m_M, M) \leq 122T,$$

then

$$\left| \zeta \left(\frac{l_M + im_M}{M} \right) \right| \leq \frac{244T}{M}.$$

Now if a is a condensation point of the sequence $(l_M + im_M)/M$ then since $\zeta(s)$ is continuous except at $s = 1$ we must have $\zeta(a) = 0$ implying the falsity of the Riemann hypothesis. Thus we have reduced the problem to the question whether for each T there is an M for which

$$B(M, T) > 122T.$$

$B(M, T)$ is primitive recursive, and the problem is therefore number-theoretic.

4. A type of problem which is not number-theoretic†.

Let us suppose that we are supplied with some unspecified means of solving number-theoretic problems; a kind of oracle as it were. We shall

† Compare Rosser [1].

not go any further into the nature of this oracle apart from saying that it cannot be a machine. With the help of the oracle we could form a new kind of machine (call them *o*-machines), having as one of its fundamental processes that of solving a given number-theoretic problem. More definitely these machines are to behave in this way. The moves of the machine are determined as usual by a table except in the case of moves from a certain internal configuration σ . If the machine is in the internal configuration σ and if the sequence of symbols marked with l is then the well-formed† formula **A**, then the machine goes into the internal configuration ρ or τ according as it is or is not true that **A** is dual. The decision as to which is the case is referred to the oracle.

These machines may be described by tables of the same kind as those used for the description of *a*-machines, there being no entries, however, for the internal configuration σ . We obtain description numbers from these tables in the same way as before. If we make the convention that, in assigning numbers to internal configurations, σ , ρ , τ are always to be q_2 , q_3 , q_4 , then the description numbers determine the behaviour of the machines uniquely.

Given any one of these machines we may ask ourselves the question whether or not it prints an infinity of figures 0 or 1; I assert that this class of problem is not number-theoretic. In view of the definition of "number theoretic problem" this means that it is not possible to construct an *o*-machine which, when supplied‡ with the description of any other *o*-machine, will determine whether that machine is *o*-circle free. The argument may be taken over directly from Turing [1], § 8. We say that a number is *o*-satisfactory if it is the description number of an *o*-circle free machine. Then, if there is an *o*-machine which will determine of any integer whether it is *o*-satisfactory, there is also an *o*-machine to calculate the values of the function $1 - \phi_n(n)$. Let $r(n)$ be the n -th *o*-satisfactory number and let $\phi_n(m)$ be the m -th figure printed by the *o*-machine whose description number is $r(n)$. This *o*-machine is circle free and there is therefore an *o*-satisfactory number K such that $\phi_K(n) = 1 - \phi_n(n)$ for all n . Putting $n = K$ yields a contradiction. This completes the proof that problems of circle freedom of *o*-machines are not number-theoretic.

Propositions of the form that an *o*-machine is *o*-circle free can always be put in the form of propositions obtained from formulae of the functional calculus of the first order by replacing *some* of the functional variables by primitive recursive relations. Compare foot-note † on page 168.

† Without real loss of generality we may suppose that **A** is always well formed.

‡ Compare Turing [1], §6, 7.

5. *Syntactical theorems as number-theoretic theorems.*

I now mention a property of number-theoretic theorems which suggests that there is reason for regarding them as of particular importance.

Suppose that we have some axiomatic system of a purely formal nature. We do not concern ourselves at all in interpretations for the formulae of this system; they are to be regarded as of interest for themselves. An example of what is in mind is afforded by the conversion calculus (§1). Every sequence of symbols "**A** conv **B**", where **A** and **B** are well formed formulae, is a formula of the axiomatic system and is provable if the W.F.F. **A** is convertible to **B**. The rules of conversion give us the rules of procedure in this axiomatic system.

Now consider a new rule of procedure which is reputed to yield only formulae provable in the original sense. We may ask ourselves whether such a rule is valid. The statement that such a rule is valid would be number-theoretic. To prove this, let us take Gödel representations for the formulae, and an enumeration of the provable formulae; let $\phi(r)$ be the G.R. of the r -th formula in the enumeration. We may suppose $\phi(r)$ to be primitive recursive if we are prepared to allow repetitions in the enumeration. Let $\psi(r)$ be the G.R. of the r -th formula obtained by the new rule, then the statement that this new rule is valid is equivalent to the assertion of

$$(r)(\exists s)[\psi(r) = \phi(s)]$$

(the domain of individuals being the natural numbers). It has been shown in §3 that such statements are number-theoretic.

It might plausibly be argued that all those theorems of mathematics which have any significance when taken alone are in effect syntactical theorems of this kind, stating the validity of certain "derived rules" of procedure. Without going so far as this, I should assert that theorems of this kind have an importance which makes it worth while to give them special consideration.

6. *Logic formulae.*

We shall call a formula **L** a *logic formula* (or, if it is clear that we are speaking of a W.F.F., simply a *logic*) if it has the property that, if **A** is a formula such that **L(A)** conv 2, then **A** is dual.

A logic formula gives us a means of satisfying ourselves of the truth of number-theoretic theorems. For to each number-theoretic proposition there corresponds a W.F.F. **A** which is dual if and only if the proposition is true. Now, if **L** is a logic and **L(A)** conv 2, then **A** is dual and we know that

the corresponding number-theoretic proposition is true. It does not follow that, if \mathbf{L} is a logic, we can use \mathbf{L} to satisfy ourselves of the truth of *any* number-theoretic theorem.

If \mathbf{L} is a logic, the set of formulae \mathbf{A} for which $\mathbf{L}(\mathbf{A}) \text{ conv 2}$ will be called the *extent* of \mathbf{L} .

It may be proved by the use of (D), (E), p. 166, that there is a formula X such that, if \mathbf{M} has a normal form, has no free variables and is not convertible to 2, then $X(\mathbf{M}) \text{ conv 1}$, but, if $\mathbf{M} \text{ conv 2}$, then $X(\mathbf{M}) \text{ conv 2}$. If \mathbf{L} is a logic, then $\lambda x. X(\mathbf{L}(x))$ is also a logic whose extent is the same as that of \mathbf{L} , and which has the property that, if \mathbf{A} has no free variables, then

$$\{\lambda x. X(\mathbf{L}(x))\}(\mathbf{A})$$

either is always convertible to 1 or to 2 or else has no normal form. A logic with this property will be said to be *standardized*.

We shall say that a logic \mathbf{L}' is *at least as complete as* a logic \mathbf{L} if the extent of \mathbf{L} is a subset of the extent of \mathbf{L}' . The logic \mathbf{L}' is *more complete than* \mathbf{L} if the extent of \mathbf{L} is a proper subset of the extent of \mathbf{L}' .

Suppose that we have an effective set of rules by which we can prove formulae to be dual; *i.e.* we have a system of symbolic logic in which the propositions proved are of the form that certain formulae are dual. Then we can find a logic formula whose extent consists of just those formulae which can be proved to be dual by the rules; that is to say, there is a rule for obtaining the logic formula from the system of symbolic logic. In fact the system of symbolic logic enables us to obtain† a computable function of positive integers whose values run through the Gödel representations of the formulae provable by means of the given rules. By the theorem of equivalence of computable and λ -definable functions, there is a formula \mathbf{J} such that $\mathbf{J}(1), \mathbf{J}(2), \dots$ are the G.R. of these formulae. Now let

$$W \rightarrow \lambda jv. \mathcal{P}(\lambda u. \delta(j(u), v), 1, I, 2).$$

Then I assert that $W(\mathbf{J})$ is a logic with the required properties. The properties of \mathcal{P} imply that $\mathcal{P}(\mathbf{C}, 1)$ is convertible to the least positive integer \mathbf{n} for which $\mathbf{C}(\mathbf{n}) \text{ conv 2}$, and has no normal form if there is no such integer. Consequently $\mathcal{P}(\mathbf{C}, 1, I, 2)$ is convertible to 2 if $\mathbf{C}(\mathbf{n}) \text{ conv 2}$ for some positive integer n , and it has no normal form otherwise. That is to say that $W(\mathbf{J}, \mathbf{A}) \text{ conv 2}$ if and only if $\delta(\mathbf{J}(\mathbf{n}), \mathbf{A}) \text{ conv 2}$, some n , *i.e.* if $\mathbf{J}(\mathbf{n}) \text{ conv } \mathbf{A}$ some n .

† Compare Turing [1], 252, second footnote, [2], 156.

There is conversely a formula W' such that, if \mathbf{L} is a logic, then $W'(\mathbf{L})$ enumerates the extent of \mathbf{L} . For there is a formula Q such that $Q(\mathbf{L}, \mathbf{A}, n)$ conv 2 if and only if $\mathbf{L}(\mathbf{A})$ is convertible to 2 in less than n steps. We then put

$$W' \rightarrow \lambda ln . \text{form} \left(\varpi \left(2, \mathfrak{P} \left(\lambda x . Q \left(l, \text{form} \left(\varpi(2, x) \right), \varpi(3, x) \right), n \right) \right) \right).$$

Of course, $W'(W(\mathbf{J}))$ normally entirely different from \mathbf{J} and $W(W'(\mathbf{L}))$ from \mathbf{L} .

In the case where we have a symbolic logic whose propositions can be interpreted as number-theoretic theorems, but are not expressed in the form of the duality of formulae, we shall again have a corresponding logic formula, but its relation to the symbolic logic is not so simple. As an example let us take the case where the symbolic logic proves that certain primitive recursive functions vanish infinitely often. As was shown in §3, we can associate with each such proposition a W.F.F. which is dual if and only if the proposition is true. When we replace the propositions of the symbolic logic by theorems on the duality of formulae in this way, our previous argument applies and we obtain a certain logic formula \mathbf{L} . However, \mathbf{L} does not determine uniquely which are the propositions provable in the symbolic logic; for it is possible that “ $\theta_1(x)$ vanishes infinitely often” and “ $\theta_2(x)$ vanishes infinitely often” are both associated with “ \mathbf{A} is dual”, and that the first of these propositions is provable in the system, but the second not. However, if we suppose that the system of symbolic logic is sufficiently powerful to be able to carry out the argument on pp. 170–171 then this difficulty cannot arise. There is also the possibility that there may be formulae in the extent of \mathbf{L} with no propositions of the form “ $\theta(x)$ vanishes infinitely often” corresponding to them. But to each such formula we can assign (by a different argument) a proposition p of the symbolic logic which is a necessary and sufficient condition for \mathbf{A} to be dual. With p is associated (in the first way) a formula \mathbf{A}' . Now \mathbf{L} can always be modified so that its extent contains \mathbf{A}' whenever it contains \mathbf{A} .

We shall be interested principally in questions of completeness. Let us suppose that we have a class of systems of symbolic logic, the propositions of these systems being expressed in a uniform notation and interpretable as number-theoretic theorems; suppose also that there is a rule by which we can assign to each proposition p of the notation a W.F.F. \mathbf{A}_p which is dual if and only if p is true, and that to each W.F.F. \mathbf{A} we can assign a propo-

sition p_A which is a necessary and sufficient condition for A to be dual. p_{A_p} is to be expected to differ from p . To each symbolic logic C we can assign two logic formulae L_C and L'_C . A formula A belongs to the extent of L_C if p_A is provable in C , while the extent of L'_C consists of all A_p , where p is provable in C . Let us say that the class of symbolic logics is complete if each true proposition is provable in one of them: let us also say that a class of logic formulae is complete if the set-theoretic sum of the extents of these logics includes all dual formulae. I assert that a necessary condition for a class of symbolic logics C to be complete is that the class of logics L_C is complete, while a sufficient condition is that the class of logics L'_C is complete. Let us suppose that the class of symbolic logics is complete; consider p_A , where A is arbitrary but dual. It must be provable in one of the systems, C say. A therefore belongs to the extent of L_C , i.e. the class of logics L_C is complete. Now suppose the class of logics L'_C to be complete. Let p be an arbitrary true proposition of the notation; A_p must belong to the extent of some L'_C , and this means that p is provable in C .

We shall say that a single logic formula L is complete if its extent includes all dual formulae; that is to say, it is complete if it enables us to prove every true number-theoretic theorem. It is a consequence of the theorem of Gödel (if suitably extended) that no logic formula is complete, and this also follows from (C), p. 165, or from the results of Turing [1], § 8, when taken in conjunction with § 3 of the present paper. The idea of completeness of a logic formula is not therefore very important, although it is useful to have a term for it.

Suppose Y to be a W.F.F. such that $Y(n)$ is a logic for each positive integer n . The formulae of the extent of $Y(n)$ are enumerated by $W(Y(n))$, and the combined extents of these logics by

$$\lambda r. W(Y(\varpi(2, r), \varpi(3, r))).$$

If we put

$$\Gamma \rightarrow \lambda y. W'(\lambda r. W(Y(\varpi(2, r), \varpi(3, r))))),$$

then $\Gamma(Y)$ is a logic whose extent is the combined extent of

$$Y(1), Y(2), Y(3), \dots$$

To each W.F.F. L we can assign a W.F.F. $V(L)$ such that a necessary and sufficient condition for L to be a logic formula is that $V(L)$ is dual. Let Nm be a W.F.F. which enumerates all formulae with normal forms

and no free variables. Then the condition for \mathbf{L} to be a logic is that $\mathbf{L}(\text{Nm}(\mathbf{r}), \mathbf{s}) \text{ conv 2}$ for all positive integers r, s , *i.e.* that

$$\lambda a. \mathbf{L}(\text{Nm}(\varpi(2, a)), \varpi(3, a))$$

is dual. We may therefore put

$$V \rightarrow \lambda a. l(\text{Nm}(\varpi(2, a)), \varpi(3, a)).$$

7. *Ordinals.*

We begin our treatment of ordinals with some brief definitions from the Cantor theory of ordinals, but for the understanding of some of the proofs a greater amount of the Cantor theory is necessary than is set out here.

Suppose that we have a class determined by the propositional function $D(x)$ and a relation $G(x, y)$ ordering its members, *i.e.* satisfying

$$\left. \begin{array}{ll} G(x, y) \& G(y, z) \supset G(x, z), & \text{(i)} \\ D(x) \& D(y) \supset G(x, y) \vee G(y, x) \vee x = y, & \text{(ii)} \\ G(x, y) \supset D(x) \& D(y), & \text{(iii)} \\ \sim G(x, x). & \text{(iv)} \end{array} \right\} \quad (7.1)$$

The class defined by $D(x)$ is then called a *series* with the ordering relation $G(x, y)$. The series is said to be *well ordered* and the ordering relation is called an *ordinal* if every sub-series which is not void has a first term, *i.e.* if

$$\begin{aligned} (D') \{ & (\exists x) (D'(x)) \& (x) (D'(x) \supset D(x)) \\ & \supset (\exists z) (y) [D'(z) \& (D'(y) \supset G(z, y) \vee z = y)] \}. \end{aligned} \quad (7.2)$$

The condition (7.2) is equivalent to another, more suitable for our purposes, namely the condition that every descending subsequence must terminate; formally

$$(x) \{ D'(x) \supset D(x) \& (\exists y) (D'(y) \& G(y, x)) \} \supset (x) (\sim D'(x)). \quad (7.3)$$

The ordering relation $G(x, y)$ is said to be similar to $G'(x, y)$ if there is a one-one correspondence between the series transforming the one relation

into the other. This is best expressed formally, thus

$$\begin{aligned}
 (\exists M) \Big[& (x) \{ D(x) \supset (\exists x') M(x, x') \} \& (x') \{ D'(x') \supset (\exists x) M(x, x') \} \\
 & \& \{ (M(x, x') \& M(x, x'')) \vee (M(x', x) \& M(x'', x)) \supset x' = x'' \} \\
 & \& \{ M(x, x') \& M(y, y') \supset (G(x, y) \supset G(x', y')) \} \Big]. \quad (7.4)
 \end{aligned}$$

Ordering relations are regarded as belonging to the same ordinal if and only if they are similar.

We wish to give names to all the ordinals, but this will not be possible until they have been restricted in some way; the class of ordinals, as at present defined, is more than enumerable. The restrictions that we actually impose are these: $D(x)$ is to imply that x is a positive integer; $D(x)$ and $G(x, y)$ are to be computable properties. Both of the propositional functions $D(x)$, $G(x, y)$ can then be described by means of a single W.F.F. Ω with the properties:

- $\Omega(m, n)$ conv 4 unless both $D(m)$ and $D(n)$ are true,
- $\Omega(m, m)$ conv 3 if $D(m)$ is true,
- $\Omega(m, n)$ conv 2 if $D(m)$, $D(n)$, $G(m, n)$, $\sim (m = n)$ are true,
- $\Omega(m, n)$ conv 1 if $D(m)$, $D(n)$, $\sim G(m, n)$, $\sim (m = n)$ are true.

In consequence of the conditions to which $D(x)$, $G(x, y)$ are subjected, Ω must further satisfy:

- (a) if $\Omega(m, n)$ is convertible to 1 or 2, then $\Omega(m, m)$ and $\Omega(n, n)$ are convertible to 3,
- (b) if $\Omega(m, m)$ and $\Omega(n, n)$ are convertible to 3, then $\Omega(m, n)$ is convertible to 1, 2, or 3,
- (c) if $\Omega(m, n)$ is convertible to 1, then $\Omega(n, m)$ is convertible to 2 and conversely,
- (d) if $\Omega(m, n)$ and $\Omega(n, p)$ are convertible to 1, then $\Omega(m, p)$ is also,
- (e) there is no sequence m_1, m_2, \dots such that $\Omega(m_{i+1}, m_i)$ conv 2 for each positive integer i ,
- (f) $\Omega(m, n)$ is always convertible to 1, 2, 3, or 4.

If a formula Ω satisfies these conditions then there are corresponding propositional functions $D(x)$, $G(x, y)$. We shall therefore say that Ω is

an *ordinal formula* if it satisfies the conditions (a)–(f). It will be seen that a consequence of this definition is that Dt is an ordinal formula; it represents the ordinal ω . The definition that we have given does not pretend to have virtues such as elegance or convenience. It has been introduced rather to fix our ideas and to show how it is possible in principle to describe ordinals by means of well formed formulae. The definitions could be modified in a number of ways. Some such modifications are quite trivial; they are typified by modifications such as changing the numbers 1, 2, 3, 4, used in the definition, to others. Two such definitions will be said to be equivalent; in general, we shall say that two definitions are equivalent if there are W.F.F. T, T' such that, if \mathbf{A} is an ordinal formula under one definition and represents the ordinal α , then $T'(\mathbf{A})$ is an ordinal formula under the second definition and represents the same ordinal; and, conversely, if \mathbf{A}' is an ordinal formula under the second definition representing α , then $T(\mathbf{A}')$ represents α under the first definition. Besides definitions equivalent in this sense to our original definition, there are a number of other possibilities open. Suppose for instance that we do not require $D(x)$ and $G(x, y)$ to be computable, but that we require only that $D(x)$ and $G(x, y) \& x < y$ are axiomatic†. This leads to a definition of an ordinal formula which is (presumably) not equivalent to the definition that we are using‡. There are numerous possibilities, and little to guide us in choosing one definition rather than another. No one of them could well be described as “wrong”; some of them may be found more valuable in applications than others, and the particular choice that we have made has been determined partly by the applications that we have in view. In the case of theorems of a negative character, one would wish to prove them for each one of the possible definitions of “ordinal formula”. This programme could, I think, be carried through for the negative results of §9, 10.

Before leaving the subject of possible ways of defining ordinal formulae, I must mention another definition due to Church and Kleene (Church and Kleene [1]). We can make use of this definition in constructing ordinal logics, but it is more convenient to use a slightly different definition which is equivalent (in the sense just described) to the Church-Kleene definition as modified in Church [4].

† To require $G(x, y)$ to be axiomatic amounts to requiring $G(x, y)$ to be computable on account of (7. 1) (ii).

‡ On the other hand, if $D(x)$ is axiomatic and $G(x, y)$ is computable in the modified sense that there is a rule for determining whether $G(x, y)$ is true which leads to a definite result in all cases where $D(x)$ and $D(y)$ are true, the corresponding definition of ordinal formula is equivalent to our definition. To give the proof would be too much of a digression. Probably other equivalences of this kind hold.

Introduce the abbreviations

$$U \rightarrow \lambda ufx . u \left(\lambda y . f \left(y(I, x) \right) \right),$$

$$\text{Suc} \rightarrow \lambda aufx . f \left(a(u, f, x) \right).$$

We define first a partial ordering relation “ $<$ ” which holds between certain pairs of W.F.F. [conditions (1)–(5)].

- (1) If \mathbf{A} conv \mathbf{B} , then $\mathbf{A} < \mathbf{C}$ implies $\mathbf{B} < \mathbf{C}$ and $\mathbf{C} < \mathbf{A}$ implies $\mathbf{C} < \mathbf{B}$.
- (2) $\mathbf{A} < \text{Suc}(\mathbf{A})$.
- (3) For any positive integers m and n , $\lambda ufx . \mathbf{R}(\mathbf{n}) < \lambda ufx . \mathbf{R}(\mathbf{m})$ implies $\lambda ufx . \mathbf{R}(\mathbf{n}) < \lambda ufx . u(\mathbf{R})$.
- (4) If $\mathbf{A} < \mathbf{B}$ and $\mathbf{B} < \mathbf{C}$, then $\mathbf{A} < \mathbf{C}$. (1)–(4) are required for any W.F.F. \mathbf{A} , \mathbf{B} , \mathbf{C} , $\lambda ufx . \mathbf{R}$.
- (5) The relation $\mathbf{A} < \mathbf{B}$ holds only when compelled to do so by (1)–(4).

We define C-K ordinal formulae by the conditions (6)–(10).

- (6) If \mathbf{A} conv \mathbf{B} and \mathbf{A} is a C-K ordinal formula, then \mathbf{B} is a C-K ordinal formula.

(7) U is a C-K ordinal formula.

(8) If \mathbf{A} is a C-K ordinal formula, then $\text{Suc}(\mathbf{A})$ is a C-K ordinal formula.

(9) If $\lambda ufx . \mathbf{R}(\mathbf{n})$ is a C-K ordinal formula and

$$\lambda ufx . \mathbf{R}(\mathbf{n}) < \lambda ufx . \mathbf{R}(S(\mathbf{n}))$$

for each positive integer n , then $\lambda ufx . u(\mathbf{R})$ is a C-K ordinal formula†.

- (10) A formula is a C-K ordinal formula only if compelled to be so by (6)–(9).

† If we also allow $\lambda ufx . u(\mathbf{R})$ to be a C-K ordinal formula when

$$\lambda ufx . \mathbf{n}(\mathbf{R}) \text{ conv } \lambda ufx . S(\mathbf{n}, \mathbf{R})$$

for all n , then the formulae for sum, product and exponentiation of C-K ordinal formulae can be much simplified. For instance, if \mathbf{A} and \mathbf{B} represent α and β , then

$$\lambda ufx . \mathbf{B}(u, f, \mathbf{A}(u, f, x))$$

represents $\alpha + \beta$. Property (6) remains true.

The representation of ordinals by formulae is described by (11)–(15).

- (11) If $\mathbf{A} \text{ conv } \mathbf{B}$ and \mathbf{A} represents α , then \mathbf{B} represents α .
- (12) U represents 1.
- (13) If \mathbf{A} represents α , then $\text{Suc}(\mathbf{A})$ represents $\alpha + 1$.
- (14) If $\lambda ufx.R(n)$ represents α_n for each positive integer n , then $\lambda ufx.u(\mathbf{R})$ represents the upper bound of the sequence $\alpha_1, \alpha_2, \alpha_3, \dots$.
- (15) A formula represents an ordinal only when compelled to do so by (11)–(14).

We denote any ordinal represented by \mathbf{A} by $\Xi_{\mathbf{A}}$ without prejudice to the possibility that more than one ordinal may be represented by \mathbf{A} . We shall write $\mathbf{A} \leq \mathbf{B}$ to mean $\mathbf{A} < \mathbf{B}$ or $\mathbf{A} \text{ conv } \mathbf{B}$.

In proving properties of C-K ordinal formulae we shall often use a kind of analogue of the principle of transfinite induction. If ϕ is some property and we have :

- (a) If $\mathbf{A} \text{ conv } \mathbf{B}$ and $\phi(\mathbf{A})$, then $\phi(\mathbf{B})$,
- (b) $\phi(U)$,
- (c) If $\phi(\mathbf{A})$, then $\phi(\text{Suc}(\mathbf{A}))$,
- (d) If $\phi(\lambda ufx.R(n))$ and $\lambda ufx.R(n) < \lambda ufx.R(S(n))$ for each positive integer n , then $\phi(\lambda ufx.u(\mathbf{R}))$;

} (7.5)

then $\phi(\mathbf{A})$ for each C-K ordinal formula \mathbf{A} . To prove the validity of this principle we have only to observe that the class of formulae \mathbf{A} satisfying $\phi(\mathbf{A})$ is one of those of which the class of C-K ordinal formulae was defined to be the smallest. We can use this principle to help us to prove :—

- (i) Every C-K ordinal formula is convertible to the form $\lambda ufx.\mathbf{B}$, where \mathbf{B} is in normal form.
- (ii) There is a method by which for any C-K ordinal formula, we can determine into which of the forms U , $\text{Suc}(\lambda ufx.\mathbf{B})$, $\lambda ufx.u(\mathbf{R})$ (where u is free in \mathbf{R}) it is convertible, and by which we can determine \mathbf{B} , \mathbf{R} . In each case \mathbf{B} , \mathbf{R} are unique apart from conversions.
- (iii) If \mathbf{A} represents any ordinal, $\Xi_{\mathbf{A}}$ is unique. If $\Xi_{\mathbf{A}}$, $\Xi_{\mathbf{B}}$ exist and $\mathbf{A} < \mathbf{B}$, then $\Xi_{\mathbf{A}} < \Xi_{\mathbf{B}}$.

(iv) If $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are C-K ordinal formulae and $\mathbf{B} < \mathbf{A}, \mathbf{C} < \mathbf{A}$, then either $\mathbf{B} < \mathbf{C}, \mathbf{C} < \mathbf{B}$, or \mathbf{B} conv \mathbf{C} .

(v) A formula \mathbf{A} is a C-K ordinal formula if:

(A) $U \leqslant \mathbf{A}$,

(B) If $\lambda ufx . u(\mathbf{R}) \leqslant \mathbf{A}$ and n is a positive integer, then

$$\lambda ufx . \mathbf{R}(\mathbf{n}) < \lambda ufx . \mathbf{R}(S(\mathbf{n})),$$

(C) For any two W.F.F. \mathbf{B}, \mathbf{C} with $\mathbf{B} < \mathbf{A}, \mathbf{C} < \mathbf{A}$ we have $\mathbf{B} < \mathbf{C}, \mathbf{C} < \mathbf{B}$, or \mathbf{B} conv \mathbf{C} , but never $\mathbf{B} < \mathbf{B}$,

(D) There is no infinite sequence $\mathbf{B}_1, \mathbf{B}_2, \dots$ for which

$$\mathbf{B}_r < \mathbf{B}_{r-1} < \mathbf{A}$$

for each r .

(vi) There is a formula H such that, if \mathbf{A} is a C-K ordinal formula, then $H(\mathbf{A})$ is an ordinal formula representing the same ordinal. $H(\mathbf{A})$ is not an ordinal formula unless \mathbf{A} is a C-K ordinal formula.

Proof of (i). Take $\phi(\mathbf{A})$ to be “ \mathbf{A} is convertible to the form $\lambda ufx . \mathbf{B}$, where \mathbf{B} is in normal form”. The conditions (a) and (b) are trivial. For (c), suppose that \mathbf{A} conv $\lambda ufx . \mathbf{B}$, where \mathbf{B} is in normal form; then

$$\text{Suc}(\mathbf{A}) \text{ conv } \lambda ufx . f(\mathbf{B})$$

and $f(\mathbf{B})$ is in normal form. For (d) we have only to show that $u(\mathbf{R})$ has a normal form, *i.e.* that \mathbf{R} has a normal form; and this is true since $\mathbf{R}(1)$ has a normal form.

Proof of (ii). Since, by hypothesis, the formula is a C-K ordinal formula we have only to perform conversions on it until it is in one of the forms described. It is not possible to convert it into two of these three forms. For suppose that $\lambda ufx . f(\mathbf{A}(u, f, x))$ conv $\lambda ufx . u(\mathbf{R})$ and is a C-K ordinal formula; it is then convertible to the form $\lambda ufx . \mathbf{B}$, where \mathbf{B} is in normal form. But the normal form of $\lambda ufx . u(\mathbf{R})$ can be obtained by conversions on \mathbf{R} , and that of $\lambda ufx . f(\mathbf{A}(u, f, x))$ by conversions on $\mathbf{A}(u, f, x)$ (as follows from Church and Rosser [1], Theorem 2); this, however, would imply that the formula in question had two normal forms, one of form $\lambda ufx . u(\mathbf{S})$ and one of form $\lambda ufx . f(\mathbf{C})$, which is impossible. Or let U conv $\lambda ufx . u(\mathbf{R})$, where \mathbf{R} is a well formed formula with u as a free variable. We may suppose \mathbf{R} to be in normal form. Now U is $\lambda ufx . u(\lambda y . f(y(I, x)))$. By

(A), p. 165, \mathbf{R} is identical with $\lambda y.f(y(I, x))$, which does not have u as a free variable. It now remains to show only that if

$\text{Suc}(\lambda ufx.\mathbf{B}) \text{ conv } \text{Suc}(\lambda ufx.\mathbf{B}')$ and $\lambda ufx.u(\mathbf{R}) \text{ conv } \lambda ufx.u(\mathbf{R}')$,
then $\mathbf{B} \text{ conv } \mathbf{B}'$ and $\mathbf{R} \text{ conv } \mathbf{R}'$.

If $\text{Suc}(\lambda ufx.\mathbf{B}) \text{ conv } \text{Suc}(\lambda ufx.\mathbf{B}')$,
then $\lambda ufx.f(\mathbf{B}) \text{ conv } \lambda ufx.f(\mathbf{B}')$;

but both of these formulae can be brought to normal form by conversions on \mathbf{B} , \mathbf{B}' and therefore $\mathbf{B} \text{ conv } \mathbf{B}'$. The same argument applies in the case in which $\lambda ufx.u(\mathbf{R}) \text{ conv } \lambda ufx.u(\mathbf{R}')$.

Proof of (iii). To prove the first half, take $\phi(\mathbf{A})$ to be “ $\Xi_{\mathbf{A}}$ is unique”. Then (7.5) (a) is trivial, and (b) follows from the fact that U is not convertible either to the form $\text{Suc}(\mathbf{A})$ or to $\lambda ufx.u(\mathbf{R})$, where \mathbf{R} has u as a free variable. For (c): $\text{Suc}(\mathbf{A})$ is not convertible to the form $\lambda ufx.u(\mathbf{R})$; the possibility that $\text{Suc}(\mathbf{A})$ represents an ordinal on account of (12) or (14) is therefore eliminated. By (13), $\text{Suc}(\mathbf{A})$ represents $\alpha' + 1$ if \mathbf{A}' represents α' and $\text{Suc}(\mathbf{A}) \text{ conv } \text{Suc}(\mathbf{A}')$. If we suppose that \mathbf{A} represents α , then \mathbf{A}, \mathbf{A}' , being C-K ordinal formulae, are convertible to the forms $\lambda ufx.\mathbf{B}, \lambda ufx.\mathbf{B}'$; but then, by (ii), $\mathbf{B} \text{ conv } \mathbf{B}'$, i.e. $\mathbf{A} \text{ conv } \mathbf{A}'$, and therefore $\alpha = \alpha'$ by the hypothesis $\phi(\mathbf{A})$. Then $\Xi_{\text{Suc}(\mathbf{A})} = \alpha' + 1$ is unique. For (d): $\lambda ufx.u(\mathbf{R})$ is not convertible to the form $\text{Suc}(\mathbf{A})$ or to U if \mathbf{R} has u as a free variable. If $\lambda ufx.u(\mathbf{R})$ represents an ordinal, it is so therefore in virtue of (14), possibly together with (11). Now, if $\lambda ufx.u(\mathbf{R}) \text{ conv } \lambda ufx.u(\mathbf{R}')$, then $\mathbf{R} \text{ conv } \mathbf{R}'$, so that the sequence $\lambda ufx.\mathbf{R}(1), \lambda ufx.\mathbf{R}(2), \dots$ in (14) is unique apart from conversions. Then, by the induction hypothesis, the sequence $\alpha_1, \alpha_2, \alpha_3, \dots$ is unique. The only ordinal that is represented by $\lambda ufx.u(\mathbf{R})$ is the upper bound of this sequence; and this is unique.

For the second half we use a type of argument rather different from our transfinite induction principle. The formulae \mathbf{B} for which $\mathbf{A} < \mathbf{B}$ form the smallest class for which:

$$\left. \begin{array}{l} \text{Suc}(\mathbf{A}) \text{ belongs to the class.} \\ \text{If } \mathbf{C} \text{ belongs to the class, then } \text{Suc}(\mathbf{C}) \text{ belongs to it.} \\ \text{If } \lambda ufx.\mathbf{R}(\mathbf{n}) \text{ belongs to the class and} \\ \qquad \qquad \qquad \lambda ufx.\mathbf{R}(\mathbf{n}) < \lambda ufx.\mathbf{R}(\mathbf{m}), \end{array} \right\} (7.6)$$

where m, n are some positive integers, then $\lambda ufx.u(\mathbf{R})$ belongs to it.

If \mathbf{C} belongs to the class and $\mathbf{C} \text{ conv } \mathbf{C}'$, then \mathbf{C}' belongs to it.

It will be sufficient to prove that the class of formulae \mathbf{B} for which either $\Xi_{\mathbf{B}}$ does not exist or $\Xi_{\mathbf{A}} < \Xi_{\mathbf{B}}$ satisfies the conditions (7.6). Now

$$\Xi_{\text{Suc}(\mathbf{A})} = \Xi_{\mathbf{A}} + 1 > \Xi_{\mathbf{A}},$$

$$\Xi_{\text{Suc}(\mathbf{C})} > \Xi_{\mathbf{C}} > \Xi_{\mathbf{A}} \text{ if } \mathbf{C} \text{ is in the class.}$$

If $\Xi_{\lambda ufx. R(n)}$ does not exist, then $\Xi_{\lambda ufx. u(R)}$ does not exist, and therefore $\lambda ufx. u(R)$ is in the class. If $\Xi_{\lambda ufx. R(n)}$ exists and is greater than $\Xi_{\mathbf{A}}$, and $\lambda ufx. R(n) < \lambda ufx. R(m)$, then

$$\Xi_{\lambda ufx. u(R)} \geq \Xi_{\lambda ufx. R(n)} > \Xi_{\mathbf{A}},$$

so that $\lambda ufx. u(R)$ belongs to the class.

Proof of (iv). We prove this by induction with respect to \mathbf{A} . Take $\phi(\mathbf{A})$ to be “whenever $\mathbf{B} < \mathbf{A}$ and $\mathbf{C} < \mathbf{A}$ then $\mathbf{B} < \mathbf{C}$ or $\mathbf{C} < \mathbf{B}$ or $\mathbf{B} \text{ conv } \mathbf{C}$ ”. $\phi(U)$ follows from the fact that we never have $\mathbf{B} < U$. If we have $\phi(\mathbf{A})$ and $\mathbf{B} < \text{Suc}(\mathbf{A})$, then either $\mathbf{B} < \mathbf{A}$ or $\mathbf{B} \text{ conv } \mathbf{A}$; for we can find \mathbf{D} such that $\mathbf{B} \leq \mathbf{D}$, and then $\mathbf{D} < \text{Suc}(\mathbf{A})$ can be proved without appealing either to (1) or (5); (4) does not apply, so we must have $\mathbf{D} \text{ conv } \mathbf{A}$. Then, if $\mathbf{B} < \text{Suc}(\mathbf{A})$ and $\mathbf{C} < \text{Suc}(\mathbf{A})$, we have four possibilities,

$$\mathbf{B} \text{ conv } \mathbf{A}, \quad \mathbf{C} \text{ conv } \mathbf{A},$$

$$\mathbf{B} \text{ conv } \mathbf{A}, \quad \mathbf{C} < \mathbf{A},$$

$$\mathbf{B} < \mathbf{A}, \quad \mathbf{C} \text{ conv } \mathbf{A},$$

$$\mathbf{B} < \mathbf{A}, \quad \mathbf{C} < \mathbf{A}.$$

In the first case $\mathbf{B} \text{ conv } \mathbf{C}$, in the second $\mathbf{C} < \mathbf{B}$, in the third $\mathbf{B} < \mathbf{C}$, and in the fourth the induction hypothesis applies.

Now suppose that $\lambda ufx. R(n)$ is a C-K ordinal formula, that

$$\lambda ufx. R(n) < \lambda ufx. R(S(n)) \text{ and } \phi(R(n)),$$

for each positive integer n , and that $\mathbf{A} \text{ conv } \lambda ufx. u(R)$. Then, if $\mathbf{B} < \mathbf{A}$, this means that $\mathbf{B} < \lambda ufx. R(n)$ for some n ; if we have also $\mathbf{C} < \mathbf{A}$, then $\mathbf{B} < \lambda ufx. R(q)$, $\mathbf{C} < \lambda ufx. R(q)$ for some q . Thus, for these \mathbf{B} and \mathbf{C} , the required result follows from $\phi(\lambda ufx. R(q))$.

Proof of (v). The conditions (C), (D) imply that the classes of interconvertible formulae \mathbf{B} , $\mathbf{B} < \mathbf{A}$ are well-ordered by the relation “ $<$ ”. We prove (v) by (ordinary) transfinite induction with respect to the order type α of the series formed by these classes; (α is, in fact, the solution of the

equation $1+\alpha = \Xi_A$, but we do not need this). We suppose then that (v) is true for all order types less than α . If $\mathbf{E} < \mathbf{A}$, then \mathbf{E} satisfies the conditions of (v) and the corresponding order type is smaller: \mathbf{E} is therefore a C-K ordinal formula. This expresses all consequences of the induction hypothesis that we need. There are three cases to consider:

(x) $\alpha = 0$.

(y) $\alpha = \beta + 1$.

(z) α is of neither of the forms (x), (y).

In case (x) we must have $\mathbf{A} \text{ conv } U$ on account of (A). In case (y) there is a formula \mathbf{D} such that $\mathbf{D} < \mathbf{A}$, and $\mathbf{B} \leq \mathbf{D}$ whenever $\mathbf{B} < \mathbf{A}$. The relation $\mathbf{D} < \mathbf{A}$ must hold in virtue of either (1), (2), (3), or (4). It cannot be in virtue of (4); for then there would be \mathbf{B} , $\mathbf{B} < \mathbf{A}$, $\mathbf{D} < \mathbf{B}$ contrary to (C), taken in conjunction with the definition of \mathbf{D} . If it is in virtue of (3), then α is the upper bound of a sequence $\alpha_1, \alpha_2, \dots$ of ordinals, which are increasing by reason of (iii) and the conditions $\lambda ufx. \mathbf{R}(\mathbf{n}) < \lambda ufx. \mathbf{R}(S(\mathbf{n}))$ in (B). This is inconsistent with $\alpha = \beta + 1$. This means that (2) applies [after we have eliminated (1) by suitable conversions on \mathbf{A}, \mathbf{D}] and we see that $\mathbf{A} \text{ conv } \text{Suc}(\mathbf{D})$; but, since $\mathbf{D} < \mathbf{A}$, \mathbf{D} is a C-K ordinal formula, and \mathbf{A} must therefore be a C-K ordinal formula by (8). Now take case (z). It is impossible for \mathbf{A} to be of the form $\text{Suc}(\mathbf{D})$, for then we should have $\mathbf{B} < \mathbf{D}$ whenever $\mathbf{B} < \mathbf{A}$, and this would mean that we had case (y). Since $U < \mathbf{A}$, there must be an \mathbf{F} such that $\mathbf{F} < \mathbf{A}$ is demonstrable either by (2) or by (3) (after a possible conversion on \mathbf{A}); it must of course be demonstrable by (3). Then \mathbf{A} is of the form $\lambda ufx. u(\mathbf{R})$. By (3), (B) we see that $\lambda ufx. \mathbf{R}(\mathbf{n}) < \mathbf{A}$ for each positive integer n ; each $\lambda ufx. \mathbf{R}(\mathbf{n})$ is therefore a C-K ordinal formula. Applying (9), (B) we see that \mathbf{A} is a C-K ordinal formula.

Proof of (vi). To prove the first half, it is sufficient to find a method whereby from a C-K ordinal formula \mathbf{A} we can find the corresponding ordinal formula Ω . For then there is a formula H_1 such that $H_1(\mathbf{a}) \text{ conv } \mathbf{p}$ if \mathbf{a} is the G.R. of \mathbf{A} and \mathbf{p} is that of Ω . H is then to be defined by

$$H \rightarrow \lambda a. \text{form} \left(H_1(\text{Gr}(a)) \right).$$

The method of finding Ω may be replaced by a method of finding $\Omega(\mathbf{m}, \mathbf{n})$, given \mathbf{A} and any two positive integers m, n . We shall arrange the method so that, whenever \mathbf{A} is not an ordinal formula, either the calculation of the values does not terminate or else the values are not consistent with

Ω being an ordinal formula. In this way we can prove the second half of (vi).

Let Ls be a formula such that $Ls(\mathbf{A})$ enumerates the classes of formulae \mathbf{B} , $\mathbf{B} < \mathbf{A}$ [i.e. if $\mathbf{B} < \mathbf{A}$ there is one and only one positive integer n for which $Ls(\mathbf{A}, n) \text{ conv } \mathbf{B}$]. Then the rule for finding the value of $\Omega(\mathbf{m}, \mathbf{n})$ is as follows:—

First determine whether $U \leq \mathbf{A}$ and whether \mathbf{A} is convertible to the form $\mathbf{r}(\text{Suc}, U)$. This terminates if \mathbf{A} is a C-K ordinal formula.

If $\mathbf{A} \text{ conv } \mathbf{r}(\text{Suc}, U)$ and either $m > r+1$ or $n > r+1$, then the value is 4. If $m < n \leq r+1$, the value is 2. If $n < m \leq r+1$, the value is 1. If $m = n \leq r+1$, the value is 3.

If \mathbf{A} is not convertible to this form, we determine whether either \mathbf{A} or $Ls(\mathbf{A}, \mathbf{m})$ is convertible to the form $\lambda ufx.u(\mathbf{R})$; and if either of them is, we verify that $\lambda ufx.\mathbf{R}(\mathbf{n}) < \lambda ufx.\mathbf{R}(S(\mathbf{n}))$. We shall eventually come to an affirmative answer if \mathbf{A} is a C-K ordinal formula.

Having checked this, we determine concerning m and n whether $Ls(\mathbf{A}, \mathbf{m}) < Ls(\mathbf{A}, \mathbf{n})$, $Ls(\mathbf{A}, \mathbf{n}) < Ls(\mathbf{A}, \mathbf{m})$, or $m = n$, and the value is to be accordingly 1, 2, or 3.

If \mathbf{A} is a C-K ordinal formula, this process certainly terminates. To see that the values so calculated correspond to an ordinal formula, and one representing $\Xi_{\mathbf{A}}$, first observe that this is so when $\Xi_{\mathbf{A}}$ is finite. In the other case (iii) and (iv) show that $\Xi_{\mathbf{B}}$ determines a one-one correspondence between the ordinals β , $1 \leq \beta \leq \Xi_{\mathbf{A}}$, and the classes of interconvertible formulae \mathbf{B} , $\mathbf{B} < \mathbf{A}$. If we take $G(m, n)$ to be $Ls(\mathbf{A}, \mathbf{m}) < Ls(\mathbf{A}, \mathbf{n})$, we see that $G(m, n)$ is the ordering relation of a series of order type† $\Xi_{\mathbf{A}}$ and on the other hand that the values of $\Omega(\mathbf{m}, \mathbf{n})$ are related to $G(m, n)$ as on p. 179.

To prove the second half suppose that \mathbf{A} is not a C-K ordinal formula. Then one of the conditions (A)–(D) in (v) must not be satisfied. If (A) is not satisfied we shall not obtain a result even in the calculation of $\Omega(1, 1)$. If (B) is not satisfied, we shall have for some positive integers p and q ,

$$Ls(\mathbf{A}, \mathbf{p}) \text{ conv } \lambda ufx.u(\mathbf{R})$$

but not $\lambda ufx.\mathbf{R}(\mathbf{q}) < \lambda ufx.\mathbf{R}(S(\mathbf{q}))$. Then the process of calculating $\Omega(\mathbf{p}, \mathbf{q})$ will not terminate. In case of failure of (C) or (D) the values of $\Omega(\mathbf{m}, \mathbf{n})$ may all be calculable, but if so conditions (a)–(f), p. 179, will be violated. Thus, if \mathbf{A} is not a C-K ordinal formula, then $H(\mathbf{A})$ is not an ordinal formula.

† The order type is β , where $1 + \beta = \Xi_{\mathbf{A}}$; but $\beta = \Xi_{\mathbf{A}}$, since $\Xi_{\mathbf{A}}$ is infinite.

I propose now to define three formulae Sum, Lim, Inf of importance in connection with ordinal formulae. Since they are comparatively simple, they will for once be given almost in full. The formula Ug is one with the property that $Ug(m)$ is convertible to the formula representing the largest odd integer dividing m : it is not given in full. P is the predecessor function; $P(S(m))$ conv m , $P(1)$ conv 1.

$$Al \rightarrow \lambda pxy . p(\lambda guv . g(v, u), \lambda uuv . u(I, v), x, y),$$

$$Hf \rightarrow \lambda m . P\left(m\left(\lambda guv . g(v, S(u)), \lambda uuv . v(I, u), 1, 2\right)\right),$$

$$Bd \rightarrow \lambda ww' aa' x . Al\left(\lambda f . w(a, a, w'(a', a', f)), x, 4\right),$$

$$\begin{aligned} \text{Sum} \rightarrow \lambda ww' pq . Bd\left(w, w', Hf(p), Hf(q), \right. \\ \left. Al\left(p, Al\left(q, w'\left(Hf(p), Hf(q)\right), 1\right), Al\left(S(q), w\left(Hf(p), Hf(q)\right), 2\right)\right)\right), \end{aligned}$$

$$\begin{aligned} \text{Lim} \rightarrow \lambda zpq . \left\{ \lambda ab . Bd\left(z(a), z(b), Ug(p), Ug(q), Al\left(Dt(a, b) + Dt(b, a), \right. \right. \right. \\ \left. \left. \left. Dt(a, b), z(a, Ug(p), Ug(q))\right)\right)\right\}(\varpi(2, p), \varpi(2, q)), \end{aligned}$$

$$Inf \rightarrow \lambda wapq . Al\left(\lambda f . w(a, p, w(a, q, f)), w(p, q), 4\right).$$

The essential properties of these formulae are described by :

$$Al(2r-1, m, n) \text{ conv } m, \quad Al(2r, m, n) \text{ conv } n,$$

$$Hf(2m) \text{ conv } m, \quad Hf(2m-1) \text{ conv } m,$$

$$Bd(\Omega, \Omega', a, a', x) \text{ conv } 4, \text{ unless both}$$

$$\Omega(a, a) \text{ conv } 3 \quad \text{and} \quad \Omega'(a', a') \text{ conv } 3,$$

it is then convertible to x .

If Ω, Ω' are ordinal formulae representing α, β respectively, then $Sum(\Omega, \Omega')$ is an ordinal formula representing $\alpha + \beta$. If Z is a W.F.F. enumerating a sequence of ordinal formulae representing a_1, a_2, \dots , then $Lim(Z)$ is an ordinal formula representing the infinite sum $a_1 + a_2 + a_3 \dots$

If Ω is an ordinal formula representing α , then $\text{Inf}(\Omega)$ enumerates a sequence of ordinal formulae representing all the ordinals less than α without repetitions other than repetitions of the ordinal 0.

To prove that there is no general method for determining about a formula whether it is an ordinal formula, we use an argument akin to that leading to the Burali-Forti paradox; but the emphasis and the conclusion are different. Let us suppose that such an algorithm is available. This enables us to obtain a recursive enumeration $\Omega_1, \Omega_2, \dots$ of the ordinal formulae in normal form. There is a formula Z such that $Z(n) \text{ conv } \Omega_n$. Now $\text{Lim}(Z)$ represents an ordinal greater than any represented by an Ω_n , and it has therefore been omitted from the enumeration.

This argument proves more than was originally asserted. In fact, it proves that, if we take any class E of ordinal formulae in normal form, such that, if A is any ordinal formula, then there is a formula in E representing the same ordinal as A , then there is no method whereby one can determine whether a W.F.F. in normal form belongs to E .

8. *Ordinal logics.*

An ordinal logic is a W.F.F. Λ such that $\Lambda(\Omega)$ is a logic formula whenever Ω is an ordinal formula.

This definition is intended to bring under one heading a number of ways of constructing logics which have recently been proposed or which are suggested by recent advances. In this section I propose to show how to obtain some of these ordinal logics.

Suppose that we have a class W of logical systems. The symbols used in each of these systems are the same, and a class of sequences of symbols called "formulae" is defined, independently of the particular system in W . The rules of procedure of a system C define an axiomatic subset of the formulae, which are to be described as the "provable formulae of C ". Suppose further that we have a method whereby, from any system C of W , we can obtain a new system C' , also in W , and such that the set of provable formulae of C' includes the provable formulae of C (we shall be most interested in the case in which they are included as a proper subset). It is to be understood that this "method" is an effective procedure for obtaining the rules of procedure of C' from those of C .

Suppose that to certain of the formulae of W we make number-theoretic theorems correspond: by modifying the definition of formula, we may suppose that this is done for all formulae. We shall say that one of the systems C is *valid* if the provability of a formula in C implies the truth of the corresponding number-theoretic theorem. Now let the relation of

C' to C be such that the validity of C implies the validity of C' , and let there be a valid system C_0 in W . Finally, suppose that, given any computable sequence C_1, C_2, \dots of systems in W , the "limit system", in which a formula is provable if and only if it is provable in one of the systems C_j , also belongs to W . These limit systems are to be regarded, not as functions of the sequence given in extension, but as functions of the rules of formation of their terms. A sequence given in extension may be described by various rules of formation, and there will be several corresponding limit systems. Each of these may be described as a limit system of the sequence.

In these circumstances we may construct an ordinal logic. Let us associate positive integers with the systems in such a way that to each C there corresponds a positive integer m_C , and that m_C completely describes the rules of procedure of C . Then there is a W.F.F. \mathbf{K} , such that

$$\mathbf{K}(m_C) \text{ conv } \mathbf{m}_{C'}$$

for each C in W , and there is a W.F.F. Θ such that, if $\mathbf{D}(r) \text{ conv } \mathbf{m}_{C_r}$ for each positive integer r , then $\Theta(\mathbf{D}) \text{ conv } \mathbf{m}_C$, where C is a limit system of C_1, C_2, \dots . With each system C of W it is possible to associate a logic formula \mathbf{L}_C : the relation between them is that, if G is a formula of W and the number-theoretic theorem corresponding to G (assumed expressed in the conversion calculus form) asserts that \mathbf{B} is dual, then $\mathbf{L}_C(\mathbf{B}) \text{ conv } 2$ if and only if G is provable in C . There is a W.F.F. \mathbf{G} such that

$$\mathbf{G}(m_C) \text{ conv } \mathbf{L}_C$$

for each C of W . Put

$$\mathbf{N} \rightarrow \lambda a. \mathbf{G}(a(\Theta, \mathbf{K}, \mathbf{m}_{C_0})).$$

I assert that $\mathbf{N}(\mathbf{A})$ is a logic formula for each C-K ordinal formula \mathbf{A} , and that, if $\mathbf{A} < \mathbf{B}$, then $\mathbf{N}(\mathbf{B})$ is more complete than $\mathbf{N}(\mathbf{A})$, provided that there are formulae provable in C' but not in C for each valid C of W .

To prove this we shall show that to each C-K ordinal formula \mathbf{A} there corresponds a unique system $C[\mathbf{A}]$ such that:

$$(i) \mathbf{A}(\Theta, \mathbf{K}, \mathbf{m}_{C_0}) \text{ conv } \mathbf{m}_{C[\mathbf{A}]},$$

and that it further satisfies:

$$(ii) C[U] \text{ is a limit system of } C_0', C_1', \dots,$$

$$(iii) C[\text{Sue}(\mathbf{A})] \text{ is } (C[\mathbf{A}])',$$

$$(iv) C[\lambda ufx. u(\mathbf{R})] \text{ is a limit system of } C[\lambda ufx. \mathbf{R}(1)], C[\lambda ufx. \mathbf{R}(2)], \dots,$$

A and $\lambda ufx.u(\mathbf{R})$ being assumed to be C-K ordinal formulae. The uniqueness of the system follows from the fact that m_C determines C completely. Let us try to prove the existence of $C[\mathbf{A}]$ for each C-K ordinal formula **A**. As we have seen (p. 182) it is sufficient to prove

- (a) $C[U]$ exists,
- (b) if $C[\mathbf{A}]$ exists, then $C[\text{Suc}(\mathbf{A})]$ exists,
- (c) if $C[\lambda ufx.\mathbf{R}(1)]$, $C[\lambda ufx.\mathbf{R}(2)]$, ... exist, then $C[\lambda ufx.u(\mathbf{R})]$ exists.

Proof of (a).

$$\{\lambda y.\mathbf{K}(y(I, \mathbf{m}_{C_0}))\}(\mathbf{n}) \text{ conv } \mathbf{K}(\mathbf{m}_{C_0}) \text{ conv } \mathbf{m}_{C_0'}$$

for all positive integers n , and therefore, by the definition of Θ , there is a system, which we call $C[U]$ and which is a limit system of C_0' , C_0'' , ..., satisfying

$$\Theta(\lambda y.\mathbf{K}(y(I, \mathbf{m}_{C_0})) \text{ conv } \mathbf{m}_{C[U]}).$$

But, on the other hand,

$$U(\Theta, \mathbf{K}, \mathbf{m}_{C_0}) \text{ conv } \Theta(\lambda y.\mathbf{K}(y(I, \mathbf{m}_{C_0}))).$$

This proves (a) and incidentally (ii).

Proof of (b).

$$\begin{aligned} \text{Suc}(\mathbf{A}, \Theta, \mathbf{K}, \mathbf{m}_{C_0}) \text{ conv } \mathbf{K}(\mathbf{A}(\Theta, \mathbf{K}, \mathbf{m}_{C_0})) \\ \text{ conv } \mathbf{K}(\mathbf{m}_{C[\mathbf{A}]}) \\ \text{ conv } \mathbf{m}_{(C[\mathbf{A}])'}. \end{aligned}$$

Hence $C[\text{Suc}(\mathbf{A})]$ exists and is given by (iii).

Proof of (c).

$$\begin{aligned} \{\{\lambda ufx.\mathbf{R}\}(\Theta, \mathbf{K}, \mathbf{m}_{C_0})\}(\mathbf{n}) \text{ conv } \{\lambda ufx.\mathbf{R}(\mathbf{n})\}(\Theta, \mathbf{K}, \mathbf{m}_{C_0}) \\ \text{ conv } \mathbf{m}_{C[\lambda ufx.\mathbf{R}(\mathbf{n})]} \end{aligned}$$

by hypothesis. Consequently, by the definition of Θ , there exists a C which is a limit system of

$$C[\lambda ufx.\mathbf{R}(1)], \quad C[\lambda ufx.\mathbf{R}(2)], \quad \dots,$$

and satisfies

$$\Theta(\{\lambda ufx . u(\mathbf{R})\}(\Theta, \mathbf{K}, \mathbf{m}_{C_0})) \text{conv } \mathbf{m}_C.$$

We define $C[\lambda ufx . u(\mathbf{R})]$ to be this C . We then have (iv) and

$$\{\lambda ufx . u(\mathbf{R})\}(\Theta, \mathbf{K}, \mathbf{m}_{C_0}) \text{conv } \Theta(\{\lambda ufx . \mathbf{R}\}(\Theta, \mathbf{K}, \mathbf{m}_{C_0}))$$

$$\text{conv } \mathbf{m}_{C[\lambda ufx . u(\mathbf{R})]}.$$

This completes the proof of the properties (i)–(iv). From (ii), (iii), (iv), the fact that C_0 is valid, and that C' is valid when C is valid, we infer that $C[\mathbf{A}]$ is valid for each C-K ordinal formula \mathbf{A} : also that there are more formulae provable in $C[\mathbf{B}]$ than in $C[\mathbf{A}]$ when $\mathbf{A} < \mathbf{B}$. The truth of our assertions regarding \mathbf{N} now follows in view of (i) and the definitions of \mathbf{N} and \mathbf{G} .

We cannot conclude that \mathbf{N} is an ordinal logic, since the formulae \mathbf{A} are C-K ordinal formulae; but the formula H enables us to obtain an ordinal logic from \mathbf{N} . By the use of the formula Gr we obtain a formula Tn such that, if \mathbf{A} has a normal form, then $Tn(\mathbf{A})$ enumerates the G.R.'s of the formulae into which \mathbf{A} is convertible. Also there is a formula Ck such that, if h is the G.R. of a formula $H(\mathbf{B})$, then $Ck(h) \text{conv } \mathbf{B}$, but otherwise $Ck(h) \text{conv } U$. Since $H(\mathbf{B})$ is an ordinal formula only if \mathbf{B} is a C-K ordinal formula, $Ck(Tn(\Omega, n))$ is a C-K ordinal formula for each ordinal formula Ω and each integer n . For many ordinal formulae it will be convertible to U , but, for suitable Ω , it will be convertible to any given C-K ordinal formula. If we put

$$\Lambda \rightarrow \lambda wa . \Gamma(\lambda n . \mathbf{N}(Ck(Tn(w, n))), a),$$

Λ is the required ordinal logic. In fact, on account of the properties of Γ , $\Lambda(\Omega, \mathbf{A})$ will be convertible to 2 if and only if there is a positive integer n such that

$$\mathbf{N}(Ck(Tn(\Omega, n)), \mathbf{A}) \text{conv } 2.$$

If $\Omega \text{conv } H(\mathbf{B})$, there will be an integer n such that $Ck(Tn(\Omega, n)) \text{conv } \mathbf{B}$, and then

$$\mathbf{N}(Ck(Tn(\Omega, n)), \mathbf{A}) \text{conv } \mathbf{N}(\mathbf{B}, \mathbf{A}).$$

For any n , $Ck(Tn(\Omega, n))$ is convertible to U or to some B , where $\Omega \text{ conv } H(B)$. Thus $\Lambda(\Omega, A) \text{ conv } 2$ if $\Omega \text{ conv } H(B)$ and $N(B, A) \text{ conv } 2$ or if $N(U, A) \text{ conv } 2$, but not in any other case.

We may now specialize and consider particular classes W of systems. First let us try to construct the ordinal logic described roughly in the introduction. For W we take the class of systems arising from the system of *Principia Mathematica*† by adjoining to it axiomatic (in the sense described on p. 167) sets of axioms‡. Gödel has shown that primitive recursive relations§ can be expressed by means of formulae in P . In fact, there is a rule whereby, given the recursion equations defining a primitive recursive relation, we can find a formula|| $\mathfrak{A}[x_0, \dots, z_0]$ such that

$$\mathfrak{A}[f^{(m_1)}0, \dots, f^{(m_r)}0]$$

is provable in P if $F(m_1, \dots, m_r)$ is true, and its negation is provable otherwise. Further, there is a method by which we can determine about a formula $\mathfrak{A}[x_0, \dots, z_0]$ whether it arises from a primitive recursive relation in this way, and by which we can find the equations which defined the relation. Formulae of this kind will be called *recursion formulae*. We shall make use of a property that they possess, which we cannot prove formally here without giving their definition in full, but which is essentially trivial. $Db[x_0, y_0]$ is to stand for a certain recursion formula such that $Db[f^{(m)}0, f^{(n)}0]$ is provable in P if $m = 2n$ and its negation is provable otherwise. Suppose that $\mathfrak{A}[x_0]$, $\mathfrak{B}[x_0]$ are two recursion formulae. Then the theorem which I am assuming is that there is a recursion relation $\mathfrak{C}_{\mathfrak{A}, \mathfrak{B}}[x_0]$ such that we can prove

$$\mathfrak{C}_{\mathfrak{A}, \mathfrak{B}}[x_0] \equiv (\exists y_0) \left((Db[x_0, y_0] \cdot \mathfrak{A}[y_0]) \vee (Db[fx_0, fy_0] \cdot \mathfrak{B}[y_0]) \right) \quad (8.1)$$

in P .

† Whitehead and Russell [1]. The axioms and rules of procedure of a similar system P will be found in a convenient form in Gödel [1], and I follow Gödel. The symbols for the natural numbers in P are $0, f0, ff0, \dots, f^{(n)}0 \dots$. Variables with the suffix “0” stand for natural numbers.

‡ It is sometimes regarded as necessary that the set of axioms used should be computable, the intention being that it should be possible to verify of a formula reputed to be an axiom whether it really is so. We can obtain the same effect with axiomatic sets of axioms in this way. In the rules of procedure describing which are the axioms, we incorporate a method of enumerating them, and we also introduce a rule that in the main part of the deduction, whenever we write down an axiom as such, we must also write down its position in the enumeration. It is possible to verify whether this has been done correctly.

§ A relation $F(m_1, \dots, m_r)$ is primitive recursive if it is a necessary and sufficient condition for the vanishing of a primitive recursive function $\phi(m_1, \dots, m_r)$.

|| Capital German letters will be used to stand for variable or undetermined formulae in P . An expression such as $\mathfrak{A}[\mathfrak{B}, \mathfrak{C}]$ stands for the result of substituting \mathfrak{B} and \mathfrak{C} for x_0 and y_0 in \mathfrak{A} .

The significant formulae in any of our extensions of P are those of the form

$$(x_0)(\exists y_0) \mathfrak{U}[x_0, y_0], \quad (8.2)$$

where $\mathfrak{U}[x_0, y_0]$ is a recursion formula, arising from the relation $R(m, n)$ let us say. The corresponding number-theoretic theorem states that for each natural number m there is a natural number n such that $R(m, n)$ is true.

The systems in W which are not valid are those in which a formula of the form (8.2) is provable, but at the same time there is a natural number, m say, such that, for each natural number n , $R(m, n)$ is false. This means to say that $\sim \mathfrak{U}[f^{(m)}0, f^{(n)}0]$ is provable for each natural number n . Since (8.2) is provable, $(\exists x_0) \mathfrak{U}[f^{(m)}0, y_0]$ is provable, so that

$$(\exists y_0) \mathfrak{U}[f^{(m)}0, y_0], \quad \sim \mathfrak{U}[f^{(m)}0, 0], \quad \sim \mathfrak{U}[f^{(m)}0, f0], \quad \dots \quad (8.3)$$

are all provable in the system. We may simplify (8.3). For a given m we may prove a formula of the form $\mathfrak{U}[f^{(m)}0, y_0] \equiv \mathfrak{B}[y_0]$ in P , where $\mathfrak{B}[x_0]$ is a recursion formula. Thus we find that a necessary and sufficient condition for a system of W to be valid is that for no recursion formula $\mathfrak{B}[x_0]$ are all of the formulae

$$(\exists x_0) \mathfrak{B}[x_0], \quad \sim \mathfrak{B}[0], \quad \sim \mathfrak{B}[f0], \quad \dots \quad (8.4)$$

provable. An important consequence of this is that, if

$$\mathfrak{U}_1[x_0], \quad \mathfrak{U}_2[x_0], \quad \dots, \quad \mathfrak{U}_n[x_0]$$

are recursion formulae, if

$$(\exists x_0) \mathfrak{U}_1[x_0] \vee \dots \vee (\exists x_0) \mathfrak{U}_n[x_0] \quad (8.5)$$

is provable in C , and C is valid, then we can prove $\mathfrak{U}_r[f^{(a)}0]$ in C for some natural numbers r, a , where $1 \leq r \leq n$. Let us define \mathfrak{D}_r to be the formula

$$(\exists x_0) \mathfrak{U}_1[x_0] \vee \dots \vee (\exists x_0) \mathfrak{U}_r[x_0]$$

and then define $\mathfrak{E}_r[x_0]$ recursively by the condition that $\mathfrak{E}_1[x_0]$ is $\mathfrak{U}_1[x_0]$ and $\mathfrak{E}_{r+1}[x_0]$ be $\mathfrak{E}_{\mathfrak{E}_r, \mathfrak{U}_{r+1}}[x_0]$. Now I say that

$$\mathfrak{D}_r \supset (\exists x_0) \mathfrak{E}_r[x_0] \quad (8.6)$$

is provable for $1 \leq r \leq n$. It is clearly provable for $r = 1$: suppose it to be provable for a given r . We can prove

$$(y_0)(\exists x_0) \text{Db}[x_0, y_0]$$

and

$$(y_0)(\exists x_0) \text{Db}[fx_0, fy_0],$$

from which we obtain

$$\mathfrak{E}_r[y_0] \supset (\exists x_0) \left((\text{Db}[x_0, y_0] \cdot \mathfrak{E}_r[y_0]) \vee (\text{Db}[fx_0, fy_0] \cdot \mathfrak{A}_{r+1}[y_0]) \right)$$

and

$$\mathfrak{A}_{r+1}[y_0] \supset (\exists x_0) \left((\text{Db}[x_0, y_0] \cdot \mathfrak{E}_r[y_0]) \vee (\text{Db}[fx_0, fy_0] \cdot \mathfrak{A}_{r+1}[y_0]) \right).$$

These together with (8.1) yield

$$(\exists y_0) \mathfrak{E}_r[y_0] \vee (\exists y_0) \mathfrak{A}_{r+1}[y_0] \supset (\exists x_0) \mathfrak{E}_{\mathfrak{F}_r, \mathfrak{A}_{r+1}}[x_0],$$

which is sufficient to prove (8.6) for $r+1$. Now, since (8.5) is provable in C , $(\exists x_0) \mathfrak{E}_n[x_0]$ must also be provable, and, since C is valid, this means that $\mathfrak{E}_n[f^{(m)}0]$ must be provable for some natural number m . From (8.1) and the definition of $\mathfrak{E}_n[x_0]$ we see that this implies that $\mathfrak{A}_r[f^{(a)}0]$ is provable for some natural numbers a and r , $1 \leq r \leq n$.

To any system C of W we can assign a primitive recursive relation $P_C(m, n)$ with the intuitive meaning “ m is the G.R. of a proof of the formula whose G.R. is n ”. We call the corresponding recursion formula $\text{Proof}_C[x_0, y_0]$ (*i.e.* $\text{Proof}_C[f^{(m)}0, f^{(n)}0]$ is provable when $P_C(m, n)$ is true, and its negation is provable otherwise). We can now explain what is the relation of a system C' to its predecessor C . The set of axioms which we adjoin to P to obtain C' consists of those adjoined in obtaining C , together with all formulae of the form

$$(\exists x_0) \text{Proof}_C[x_0, f^{(m)}0] \supset \mathfrak{F}, \quad (8.7)$$

where m is the G.R. of \mathfrak{F} .

We want to show that a contradiction can be obtained by assuming C' to be invalid but C to be valid. Let us suppose that a set of formulae of the form (8.4) is provable in C' . Let $\mathfrak{A}_1, \mathfrak{A}_2, \dots, \mathfrak{A}_k$ be those axioms of C' of the form (8.7) which are used in the proof of $(\exists x_0) \mathfrak{B}[x_0]$. We may suppose that none of them is provable in C . Then by the deduction theorem we see that

$$(\mathfrak{A}_1, \mathfrak{A}_2, \dots, \mathfrak{A}_k) \supset (\exists x_0) \mathfrak{B}[x_0] \quad (8.8)$$

o 2

is provable in C . Let \mathfrak{A}_l be $(\exists x_0) \text{Proof}_C[x_0, f^{(m_l)} 0] \supset \mathfrak{F}_l$. Then from (8.8) we find that

$$(\exists x_0) \text{Proof}_C[x_0, f^{(m_1)} 0] \vee \dots \vee (\exists x_0) \text{Proof}_C[x_0, f^{(m_k)} 0] \vee (\exists x_0) \mathfrak{B}[x_0]$$

is provable in C . It follows from a result which we have just proved that either $\mathfrak{B}[f^{(c)} 0]$ is provable for some natural number c , or else $\text{Proof}_C[f^{(n)} 0, f^{(m)} 0]$ is provable in C for some natural number u and some l , $1 \leq l \leq k$: but this would mean that \mathfrak{F}_l is provable in C (this is one of the points where we assume the validity of C) and therefore also in C' , contrary to hypothesis. Thus $\mathfrak{B}[f^{(c)} 0]$ must be provable in C' ; but we are also assuming $\sim \mathfrak{B}[f^{(c)} 0]$ to be provable in C' . There is therefore a contradiction in C' . Let us suppose that the axioms $\mathfrak{A}_1', \dots, \mathfrak{A}_{k'}'$, of the form (8.7), when adjoined to C are sufficient to obtain the contradiction and that none of these axioms is that provable in C . Then

$$\sim \mathfrak{A}_1' \vee \sim \mathfrak{A}_2' \vee \dots \vee \sim \mathfrak{A}_{k'}'$$

is provable in C , and if \mathfrak{A}_l' is $(\exists x_0) \text{Proof}_C[x_0, f^{(m_l)} 0] \supset \mathfrak{F}_l'$ then

$$(\exists x_0) \text{Proof}_C[x_0, f^{(m_1)} 0] \vee \dots \vee (\exists x_0) \text{Proof}_C[x_0, f^{(m_{k'})} 0]$$

is provable in C . But, by repetition of a previous argument, this means that \mathfrak{A}_l' is provable for some l , $1 \leq l \leq k'$, contrary to hypothesis. This is the required contradiction.

We may now construct an ordinal logic in the manner described on pp. 190–193. We shall, however, carry out the construction in rather more detail, and with some modifications appropriate to the particular case. Each system C of our set W may be described by means of a W.F.F. M_C which enumerates the G.R.'s of the axioms of C . There is a W.F.F. E such that, if a is the G.R. of some proposition \mathfrak{F} , then $E(M_C, a)$ is convertible to the G.R. of

$$(\exists x_0) \text{Proof}_C[x_0, f^{(a)} 0] \supset \mathfrak{F}.$$

If a is not the G.R. of any proposition in P , then $E(M_C, a)$ is to be convertible to the G.R. of $0 = 0$. From E we obtain a W.F.F. K such that $K(M_C, 2n+1) \text{conv } M_C(n)$, $K(M_C, 2n) \text{conv } E(M_C, n)$. The successor system C' is defined by $K(M_C) \text{conv } M_{C'}$. Let us choose a formula G such that $G(M_C, A) \text{conv } 2$ if and only if the number-theoretic theorem equivalent to “ A is dual” is provable in C . Then we define Λ_P by

$$\Lambda_P \rightarrow \lambda w a . \Gamma \left(\lambda y . G \left(\text{Ck} \left(\text{Tn}(w, y), \lambda m n . m(\varpi(2, n), \varpi(3, n)), K, M_P \right) \right), a \right).$$

This is an ordinal logic provided that P is valid.

Another ordinal logic of this type has in effect been introduced by Church †. Superficially this ordinal logic seems to have no more in common with Λ_P than that they both arise by the method which we have described, which uses C-K ordinal formulae. The initial systems are entirely different. However, in the relation between C and C' there is an interesting analogy. In Church's method the step from C to C' is performed by means of subsidiary axioms of which the most important (Church [2], p. 88, I_m) is almost a direct translation into his symbolism of the rule that we may take any formula of the form (8.4) as an axiom. There are other extra axioms, however, in Church's system, and it is therefore not unlikely that it is in some respects more complete than Λ_P .

There are other types of ordinal logic, apparently quite unrelated to the type that we have so far considered. I have in mind two types of ordinal logic, both of which can be best described directly in terms of ordinal formulae without any reference to C-K ordinal formulae. I shall describe here a specimen Λ_H of one of these types of ordinal logic. Ordinal logics of this kind were first considered by Hilbert (Hilbert [1], 183ff), and have also been used by Tarski (Tarski [1], 395ff); see also Gödel [1], foot-note 48^a.

Suppose that we have selected a particular ordinal formula Ω . We shall construct a modification P_Ω of the system P of Gödel (see foot-note † on p. 193. We shall say that a natural number n is a *type* if it is either even or $2p-1$, where $\Omega(p, p) \text{ conv } 3$. The definition of a variable in P is to be modified by the condition that the only admissible subscripts are to be the types in our sense. Elementary expressions are then defined as in P : in particular the definition of an elementary expression of type 0 is unchanged. An elementary formula is defined to be a sequence of symbols of the form $\mathfrak{U}_m \mathfrak{U}_n$, where $\mathfrak{U}_m, \mathfrak{U}_n$ are elementary expressions of types m, n satisfying one of the conditions (a), (b), (c).

- (a) m and n are both even and m exceeds n ,
- (b) m is odd and n is even,
- (c) $m = 2p-1, n = 2q-1$, and $\Omega(p, q) \text{ conv } 2$.

With these modifications the formal development of P_Ω is the same as that of P . We want, however, to have a method of associating number-theoretic theorems with certain of the formulae of P_Ω . We cannot take over directly the association which we used in P . Suppose that G is a

† In outline Church [1], 279–280. In greater detail Church [2], Chap. X.

formula in P interpretable as a number-theoretic theorem in the way described in the course of constructing Λ_P (p. 193). Then, if every type suffix in G is doubled, we shall obtain a formula in P_Ω which is to be interpreted as the same number-theoretic theorem. By the method of §6 we can now obtain from P_Ω a formula L_Ω which is a logic formula if P_Ω is valid; in fact, given Ω there is a method of obtaining L_Ω , so that there is a formula Λ_H such that $\Lambda_H(\Omega) \text{ conv } L_\Omega$ for each ordinal formula Ω .

Having now familiarized ourselves with ordinal logics by means of these examples we may begin to consider general questions concerning them.

9. Completeness questions.

The purpose of introducing ordinal logics was to avoid as far as possible the effects of Gödel's theorem. It is a consequence of this theorem, suitably modified, that it is impossible to obtain a complete logic formula, or (roughly speaking now) a complete system of logic. We were able, however, from a given system to obtain a more complete one by the adjunction as axioms of formulae, seen intuitively to be correct, but which the Gödel theorem shows are unprovable† in the original system; from this we obtained a yet more complete system by a repetition of the process, and so on. We found that the repetition of the process gave us a new system for each C-K ordinal formula. We should like to know whether this process suffices, or whether the system should be extended in other ways as well. If it were possible to determine about a W.F.F. in normal form whether it was an ordinal formula, we should know for certain that it was necessary to make extensions in other ways. In fact for any ordinal formula Λ it would then be possible to find a single logic formula L such that, if $\Lambda(\Omega, A) \text{ conv } 2$ for some ordinal formula Ω , then $L(A) \text{ conv } 2$. Since L must be incomplete, there must be formulae A for which $\Lambda(\Omega, A)$ is not convertible to 2 for any ordinal formula Ω . However, in view of the fact, proved in §7, that there is no method of determining about a formula in normal form whether it is an ordinal formula, the case does not arise, and there is still a possibility that some ordinal logics may be complete in some sense. There is a quite natural way of defining completeness.

Definition of completeness of an ordinal logic. We say that an ordinal logic Λ is complete if corresponding to each dual formula A there is an ordinal formula Ω_A such that $\Lambda(\Omega_A, A) \text{ conv } 2$.

† In the case of P we adjoined all of the axioms $(\exists x_0) \text{ Proof}[x_0, f^{(m)} 0] \supset \mathfrak{F}$, where m is the G.R. of \mathfrak{F} ; the Gödel theorem shows that *some* of them are unprovable in P .

As has been explained in § 2, the reference in the definition to the existence of Ω_A for each A is to be understood in the same naïve way as any reference to existence in mathematics.

There is room for modification in this definition: we might require that there is a formula X such that $X(A) \text{ conv } \Omega_A$, $X(A)$ being an ordinal formula whenever A is dual. There is no need, however, to discuss the relative merits of these two definitions, because in all cases in which we prove an ordinal logic to be complete we shall prove it to be complete even in the modified sense; but in cases in which we prove an ordinal logic to be incomplete, we use the definition as it stands.

In the terminology of § 6, Λ is complete if the class of logics $\Lambda(\Omega)$ is complete when Ω runs through all ordinal formulae.

There is another completeness property which is related to this one. Let us for the moment describe an ordinal logic Λ as *all inclusive* if to each logic formula L there corresponds an ordinal formula $\Omega_{(L)}$ such that $\Lambda(\Omega_{(L)})$ is as complete as L . Clearly every all inclusive ordinal logic is complete; for, if A is dual, then $\delta(A)$ is a logic with A in its extent. But, if Λ is complete and

$$Ai \rightarrow \lambda kw. \Gamma \left(\lambda ra. \delta \left(4, \delta \left(2, k \left(w, V \left(\text{Nm}(r) \right) \right) \right) + \delta \left(2, \text{Nm}(r, a) \right) \right) \right),$$

then $Ai(\Lambda)$ is an all inclusive ordinal logic. For, if A is in the extent of $\Lambda(\Omega_A)$ for each A , and we put $\Omega_{(L)} \rightarrow \Omega_{V(L)}$, then I say that, if B is in the extent of L , it must be in the extent of $Ai(\Lambda, \Omega_{(L)})$. In fact, we see that $Ai(\Lambda, \Omega_{V(L)}, B)$ is convertible to

$$\Gamma \left(\lambda ra. \delta \left(4, \delta \left(2, \Lambda \left(\Omega_{V(L)}, V \left(\text{Nm}(r) \right) \right) \right) + \delta \left(2, \text{Nm}(r, a) \right) \right), B \right).$$

For suitable n , $\text{Nm}(n) \text{ conv } L$ and then

$$\Lambda \left(\Omega_{V(L)}, V \left(\text{Nm}(n) \right) \right) \text{ conv } 2,$$

$$\text{Nm}(n, B) \text{ conv } 2,$$

and therefore, by the properties of Γ and δ

$$Ai(\Lambda, \Omega_{V(L)}, B) \text{ conv } 2.$$

Conversely $Ai(\Lambda, \Omega_{V(L)}, B)$ can be convertible to 2 only if both $\text{Nm}(n, B)$

and $\Lambda(\Omega_{V(L)}, V(Nm(n)))$ are convertible to 2 for some positive integer n ; but, if $\Lambda(\Omega_{V(L)}, V(Nm(n))) \text{ conv } 2$, then $Nm(n)$ must be a logic, and, since $Nm(n, B) \text{ conv } 2$, B must be dual.

It should be noticed that our definitions of completeness refer only to number-theoretic theorems. Although it would be possible to introduce formulae analogous to ordinal logics which would prove more general theorems than number-theoretic ones, and have a corresponding definition of completeness, yet, if our theorems are too general, we shall find that our (modified) ordinal logics are never complete. This follows from the argument of §4. If our "oracle" tells us, not whether any given number-theoretic statement is true, but whether a given formula is an ordinal formula, the argument still applies, and we find that there are classes of problem which cannot be solved by a uniform process even with the help of this oracle. This is equivalent to saying that there is no ordinal logic of the proposed modified type which is complete with respect to these problems. This situation becomes more definite if we take formulae satisfying conditions (a)–(e), (f') (as described at the end of §12) instead of ordinal formulae; it is then not possible for the ordinal logic to be complete with respect to any class of problems more extensive than the number-theoretic problems.

We might hope to obtain some intellectually satisfying system of logical inference (for the proof of number-theoretic theorems) with some ordinal logic. Gödel's theorem shows that such a system cannot be wholly mechanical; but with a complete ordinal logic we should be able to confine the non-mechanical steps entirely to verifications that particular formulae are ordinal formulae.

We might also expect to obtain an interesting classification of number-theoretic theorems according to "depth". A theorem which required an ordinal α to prove it would be deeper than one which could be proved by the use of an ordinal β less than α . However, this presupposes more than is justified. We now define

Invariance of ordinal logics. An ordinal logic Λ is said to be *invariant up to* an ordinal α if, whenever Ω, Ω' are ordinal formulae representing the same ordinal less than α , the extent of $\Lambda(\Omega)$ is identical with the extent of $\Lambda(\Omega')$. An ordinal logic is *invariant* if it is invariant up to each ordinal represented by an ordinal formula.

Clearly the classification into depths presupposes that the ordinal logic used is invariant.

Among the questions that we should now like to ask are

(a) Are there any complete ordinal logics?

(b) Are there any complete invariant ordinal logics?

To these we might have added “are all ordinal logics complete?”; but this is trivial; in fact, there are ordinal logics which do not suffice to prove any number-theoretic theorems whatever.

We shall now show that (a) must be answered affirmatively. In fact, we can write down a complete ordinal logic at once. Put

$$\text{Od} \rightarrow \lambda a. \{ \lambda fmn. \text{Dt}(f(m), f(n)) \} \left(\lambda s. \mathcal{P}(\lambda r. r(I, a(s)), 1, s) \right)$$

and

$$\text{Comp} \rightarrow \lambda wa. \delta(w, \text{Od}(a)).$$

I shall show that Comp is a complete ordinal logic.

For if, $\text{Comp}(\Omega, A)$ conv 2, then

$$\Omega \text{ conv Od}(A)$$

$$\text{conv } \lambda mn. \text{Dt}(\mathcal{P}(\lambda r. r(I, A(m)), 1, m), \mathcal{P}(\lambda r. r(I, A(n)), 1, n)).$$

$\Omega(m, n)$ has a normal form if Ω is an ordinal formula, so that then

$$\mathcal{P}(\lambda r. r(I, A(m)), 1)$$

has a normal form; this means that $r(I, A(m))$ conv 2 some r , i.e. $A(m)$ conv 2. Thus, if $\text{Comp}(\Omega, A)$ conv 2 and Ω is an ordinal formula, then A is dual. Comp is therefore an ordinal logic. Now suppose conversely that A is dual. I shall show that $\text{Od}(A)$ is an ordinal formula representing the ordinal ω . For

$$\mathcal{P}(\lambda r. r(I, A(m)), 1, m) \text{ conv } \mathcal{P}(\lambda r. r(I, 2), 1, m)$$

$$\text{conv } 1(m) \text{ conv } m,$$

$$\text{Od}(A, m, n) \text{ conv } \text{Dt}(m, n),$$

i.e. $\text{Od}(A)$ is an ordinal formula representing the same ordinal as Dt . But

$$\text{Comp}(\text{Od}(A), A) \text{ conv } \delta(\text{Od}(A), \text{Od}(A)) \text{ conv 2.}$$

This proves the completeness of Comp.

Of course Comp is not the kind of complete ordinal logic that we should really wish to use. The use of Comp does not make it any easier to see that **A** is dual. In fact, if we really want to use an ordinal logic a proof, of completeness for that particular ordinal logic will be of little value; the ordinals given by the completeness proof will not be ones which can easily be seen intuitively to be ordinals. The only value in a completeness proof of this kind would be to show that, if any objection is to be raised against an ordinal logic, it must be on account of something more subtle than incompleteness.

The theorem of completeness is also unexpected in that the ordinal formulae used are all formulae representing ω . This is contrary to our intentions in constructing Λ_P for instance; implicitly we had in mind large ordinals expressed in a simple manner. Here we have small ordinals expressed in a very complex and artificial way.

Before trying to solve the problem (b), let us see how far Λ_P and Λ_H are invariant. We should certainly not expect Λ_P to be invariant, since the extent of $\Lambda_P(\Omega)$ will depend on whether Ω is convertible to a formula of the form $H(\mathbf{A})$: but suppose that we call an ordinal logic Λ "C-K invariant up to α " if the extent of $\Lambda(H(\mathbf{A}))$ is the same as the extent of $\Lambda(H(\mathbf{B}))$ whenever **A** and **B** are C-K ordinal formulae representing the same ordinal less than α . How far is Λ_P C-K invariant? It is not difficult to see that it is C-K invariant up to any finite ordinal, that is to say up to ω . It is also C-K invariant up to $\omega+1$, as follows from the fact that the extent of

$$\Lambda_P(H(\lambda ufx \cdot u(\mathbf{R})))$$

is the set-theoretic sum of the extents of

$$\Lambda_P(H(\lambda ufx \cdot \mathbf{R}(1))), \quad \Lambda_P(H(\lambda ufx \cdot \mathbf{R}(2))), \quad \dots$$

However, there is no obvious reason for believing that it is C-K invariant up to $\omega+2$, and in fact it is demonstrable that this is not the case (see the end of this section). Let us find out what happens if we try to prove that the extent of

$$\Lambda_P(H(\text{Suc}(\lambda ufx \cdot u(\mathbf{R}_1))))$$

is the same as the extent of

$$\Lambda_P(H(\text{Suc}(\lambda ufx \cdot u(\mathbf{R}_2)))),$$

where $\lambda ufx.u(\mathbf{R}_1)$ and $\lambda ufx.u(\mathbf{R}_2)$ are two C-K ordinal formulae representing ω . We should have to prove that a formula interpretable as a number-theoretic theorem is provable in $C[\text{Suc}(\lambda ufx.u(\mathbf{R}_1))]$ if, and only if, it is provable in $C[\text{Suc}(\lambda ufx.u(\mathbf{R}_2))]$. Now $C[\text{Suc}(\lambda ufx.u(\mathbf{R}_1))]$ is obtained from $C[\lambda ufx.u(\mathbf{R}_1)]$ by adjoining all axioms of the form

$$(\exists x_0) \text{Proof}_{C[\lambda ufx.u(\mathbf{R}_1)]}[x_0, f^{(m)}0] \supset \mathfrak{F}, \quad (9.1)$$

where m is the G.R. of \mathfrak{F} , and $C[\text{Suc}(\lambda ufx.u(\mathbf{R}_2))]$ is obtained from $C[\lambda ufx.u(\mathbf{R}_2)]$ by adjoining all axioms of the form

$$(\exists x_0) \text{Proof}_{C[\lambda ufx.u(\mathbf{R}_2)]}[x_0, f^{(m)}0] \supset \mathfrak{F}. \quad (9.2)$$

The axioms which must be adjoined to P to obtain $C[\lambda ufx.u(\mathbf{R}_1)]$ are essentially the same as those which must be adjoined to obtain the system $C[\lambda ufx.u(\mathbf{R}_2)]$: however the *rules of procedure which have to be applied before these axioms can be written down are in general quite different in the two cases*. Consequently (9.1) and (9.2) are quite different axioms, and there is no reason to expect their consequences to be the same. A proper understanding of this will make our treatment of question (b) much more intelligible. See also footnote \ddagger on page 193.

Now let us turn to Λ_H . This ordinal logic is invariant. Suppose that Ω , Ω' represent the same ordinal, and suppose that we have a proof of a number-theoretic theorem G in P_Ω . The formula expressing the number-theoretic theorem does not involve any odd types. Now there is a one-one correspondence between the odd types such that if $2m-1$ corresponds to $2m'-1$ and $2n-1$ to $2n'-1$ then $\Omega(m, n) \text{ conv } 2$ implies $\Omega'(m', n') \text{ conv } 2$. Let us modify the odd type-subscripts occurring in the proof of G , replacing each by its mate in the one-one correspondence. There results a proof in $P_{\Omega'}$ with the same end formula G . That is to say that if G is provable in P_Ω it is provable in $P_{\Omega'}$. Λ_H is invariant.

The question (b) must be answered negatively. Much more can be proved, but we shall first prove an even weaker result which can be established very quickly, in order to illustrate the method.

I shall prove that an ordinal logic Λ cannot be invariant and have the property that the extent of $\Lambda(\Omega)$ is a strictly increasing function of the ordinal represented by Ω . Suppose that Λ has these properties; then we shall obtain a contradiction. Let \mathbf{A} be a W.F.F. in normal form and without free variables, and consider the process of carrying out conversions on $\mathbf{A}(1)$ until we have shown it convertible to 2, then converting $\mathbf{A}(2)$ to 2, then $\mathbf{A}(3)$ and so on: suppose that after r steps we are still performing the

conversion on $\mathbf{A}(\mathbf{m}_r)$. There is a formula Jh such that $Jh(\mathbf{A}, r) \text{ conv } \mathbf{m}_r$, for each positive integer r . Now let Z be a formula such that, for each positive integer n , $Z(\mathbf{n})$ is an ordinal formula representing ω^n , and suppose \mathbf{B} to be a member of the extent of $\Lambda(\text{Suc}(\text{Lim}(Z)))$ but not of the extent of $\Lambda(\text{Lim}(Z))$. Put

$$\mathbf{K}^* \rightarrow \lambda a . \Lambda \left(\text{Suc} \left(\text{Lim} \left(\lambda r . Z(Jh(a, r)) \right) \right), \mathbf{B} \right);$$

then \mathbf{K}^* is a complete logic. For, if \mathbf{A} is dual, then

$$\text{Suc} \left(\text{Lim} \left(\lambda r . Z(Jh(\mathbf{A}, r)) \right) \right)$$

represents the ordinal $\omega^\omega + 1$, and therefore $\mathbf{K}^*(\mathbf{A}) \text{ conv } 2$; but, if $\mathbf{A}(\mathbf{c})$ is not convertible to 2, then

$$\text{Suc} \left(\text{Lim} \left(\lambda r . Z(Jh(\mathbf{A}, r)) \right) \right)$$

represents an ordinal not exceeding $\omega^\omega + 1$, and $\mathbf{K}^*(\mathbf{A})$ is therefore not convertible to 2. Since there are no complete logic formulae, this proves our assertion.

We may now prove more powerful results.

Incompleteness theorems. (A) If an ordinal logic Λ is invariant up to an ordinal α , then for any ordinal formula Ω representing an ordinal β , $\beta < \alpha$, the extent of $\Lambda(\Omega)$ is contained in the (set-theoretic) sum of the extents of the logics $\Lambda(\mathbf{P})$, where \mathbf{P} is finite.

(B) If an ordinal logic Λ is C-K invariant up to an ordinal α , then for any C-K ordinal formula \mathbf{A} representing an ordinal β , $\beta < \alpha$, the extent of $\Lambda(H(\mathbf{A}))$ is contained in the (set-theoretic) sum of the extents of the logics $\Lambda(H(\mathbf{F}))$, where \mathbf{F} is a C-K ordinal formula representing an ordinal less than ω^2 .

Proof of (A). It is sufficient to prove that, if Ω represents an ordinal γ , $\omega \leq \gamma < \alpha$, then the extent of $\Lambda(\Omega)$ is contained in the set-theoretic sum of the extents of the logics $\Lambda(\Omega')$, where Ω' represents an ordinal less than γ . The ordinal γ must be of the form $\gamma_0 + \rho$, where ρ is finite and represented by \mathbf{P} say, and γ_0 is not the successor of any ordinal and is not less than ω . There are two cases to consider; $\gamma_0 = \omega$ and $\gamma_0 \geq 2\omega$. In each of them we shall obtain a contradiction from the assumption that there is a W.F.F.

B such that $\Lambda(\Omega, B)$ conv 2 whenever Ω represents γ , but is not convertible to 2 if Ω represents a smaller ordinal. Let us take first the case $\gamma_0 \geq 2\omega$. Suppose that $\gamma_0 = \omega + \gamma_1$, and that Ω_1 is an ordinal formula representing γ_1 . Let **A** be any W.F.F. with a normal form and no free variables, and let **Z** be the class of those positive integers which are exceeded by all integers n for which **A**(n) is not convertible to 2. Let **E** be the class of integers $2p$ such that $\Omega(p, n)$ conv 2 for some n belonging to **Z**. The class **E**, together with the class **Q** of all odd integers, is constructively enumerable. It is evident that the class can be enumerated with repetitions, and since it is infinite the required enumeration can be obtained by striking out the repetitions. There is, therefore, a formula **En** such that **En**(Ω, A, r) runs through the formulae of the class **E** + **Q** without repetitions as r runs through the positive integers. We define

$$Rt \rightarrow \lambda wamn. \text{Sum} \left(Dt, w, \text{En} (w, a, m), \text{En}(w, a, n) \right).$$

Then $Rt(\Omega_1, A)$ is an ordinal formula which represents γ_0 if **A** is dual, but a smaller ordinal otherwise. In fact

$$Rt(\Omega_1, A, m, n) \text{ conv } \{\text{Sum}(Dt, \Omega_1)\}(\text{En}(\Omega_1, A, m), \text{En}(\Omega_1, A, n)).$$

Now, if **A** is dual, **E** + **Q** includes all integers m for which

$$\{\text{Sum}(Dt, \Omega_1)\}(m, m) \text{ conv 3}.$$

(This depends on the particular form that we have chosen for the formula **Sum**.) Putting “ $\text{En}(\Omega_1, A, p) \text{ conv } q$ ” for $M(p, q)$, we see that condition (7.4) is satisfied, so that $Rt(\Omega_1, A)$ is an ordinal formula representing γ_0 . But, if **A** is not dual, the set **E** + **Q** consists of all integers m for which

$$\{\text{Sum}(Dt, \Omega_1)\}(m, r) \text{ conv 2},$$

where r depends only on **A**. In this case $Rt(\Omega_1, A)$ is an ordinal formula representing the same ordinal as $\text{Inf}(\text{Sum}(Dt, \Omega_1), r)$, and this is smaller than γ_0 . Now consider **K**:

$$K \rightarrow \lambda a. \Lambda \left(\text{Sum} \left(Rt(\Omega_1, A), P \right), B \right).$$

If **A** is dual, **K**(**A**) is convertible to 2 since $\text{Sum}(Rt(\Omega_1, A), P)$ represents γ . But, if **A** is not dual, it is not convertible to 2, since $\text{Sum}(Rt(\Omega_1, A), P)$ then represents an ordinal smaller than γ . In **K** we therefore have a complete logic formula, which is impossible.

Now we take the case $\gamma_0 = \omega$. We introduce a W.F.F. **Mg** such that if n is the D.N. of a computing machine \mathcal{M} , and if by the m -th complete

configuration of \mathcal{M} the figure 0 has been printed, then $Mg(n, m)$ is convertible to $\lambda pq. Al(4(P, 2p+2q), 3, 4)$ (which is an ordinal formula representing the ordinal 1), but if 0 has not been printed it is convertible to $\lambda pq. p(q, I, 4)$ (which represents 0). Now consider

$$M \rightarrow \lambda n. \Lambda \left(\text{Sum} \left(\text{Lim} \left(Mg(n) \right), P \right), B \right).$$

If the machine never prints 0, then $\text{Lim}(\lambda r. Mg(n, r))$ represents ω and $\text{Sum}(\text{Lim}(Mg(n)), P)$ represents γ . This means that $M(n)$ is convertible to 2. If, however, \mathcal{M} never prints 0, $\text{Sum}(\text{Lim}(Mg(n)), P)$ represents a finite ordinal and $M(n)$ is not convertible to 2. In M we therefore have means of determining about a machine whether it ever prints 0, which is impossible† (Turing [1], §8). This completes the proof of (A).

Proof of (B). It is sufficient to prove that, if G represents an ordinal γ , $\omega^2 \leq \gamma < \alpha$, then the extent of $\Lambda(H(G))$ is included in the set-theoretic sum of the extents of $\Lambda(H(G))$, where G represents an ordinal less than γ . We obtain a contradiction from the assumption that there is a formula B which is in the extent of $\Lambda(H(G))$ if G represents γ , but not if it represents any smaller ordinal. The ordinal γ is of the form $\delta + \omega^2 + \xi$, where $\xi < \omega^2$. Let D be a C-K ordinal formula representing δ and $\lambda ufx. Q(u, f, A(u, f, x))$ one representing $\alpha + \xi$ whenever A represents α .

We now define a formula Hg . Suppose that A is a W.F.F. in normal form and without free variables; consider the process of carrying out conversions on $A(1)$ until it is brought into the form 2, then converting $A(2)$ to 2, then $A(3)$, and so on. Suppose that at the r -th step of this process we are doing the n_r -th step in the conversion of $A(m_r)$. Thus, for instance, if A is not convertible to 2, m_r can never exceed 3. Then $Hg(A, r)$ is to be convertible to $\lambda f. f(m_r, n_r)$ for each positive integer r . Put

$$Sq \rightarrow \lambda dmn. n \left(\text{Suc}, m \left(\lambda aufx. u \left(\lambda y. y \left(\text{Suc}, a(u, f, x) \right) \right), d(u, f, x) \right) \right),$$

$$M \rightarrow \lambda aufx. Q(u, f, u \left(\lambda y. Hg(a, y, Sq(D)) \right)),$$

$$K_1 \rightarrow \lambda a. \Lambda(M(a), B),$$

† This part of the argument can equally well be based on the impossibility of determining about two W.F.F. whether they are interconvertible. (Church [8], 363.)

then I say that \mathbf{K}_1 is a complete logic formula. $\text{Sq}(\mathbf{D}, \mathbf{m}, \mathbf{n})$ is a C-K ordinal formula representing $\delta + m\omega + n$, and therefore $\text{Hg}(\mathbf{A}, \mathbf{r}, \text{Sq}(\mathbf{D}))$ represents an ordinal ζ , which increases steadily with increasing r , and tends to the limit $\delta + \omega^2$ if \mathbf{A} is dual. Further

$$\text{Hg}(\mathbf{A}, \mathbf{r}, \text{Sq}(\mathbf{D})) < \text{Hg}(\mathbf{A}, S(\mathbf{r}), \text{Sq}(\mathbf{D}))$$

for each positive integer r . Therefore $\lambda ufx.u(\lambda y.\text{Hg}(\mathbf{A}, y, \text{Sq}(\mathbf{D})))$ is a C-K ordinal formula and represents the limit of the sequence $\zeta_1, \zeta_2, \zeta_3, \dots$. This is $\delta + \omega^2$ if \mathbf{A} is dual, but a smaller ordinal otherwise. Likewise $\mathbf{M}(\mathbf{A})$ represents γ if \mathbf{A} is dual, but is a smaller ordinal otherwise. The formula \mathbf{B} therefore belongs to the extent of $\Lambda(H(\mathbf{M}(\mathbf{A})))$ if and only if \mathbf{A} is dual, and this implies that \mathbf{K}_1 is a complete logic formula, as was asserted. But this is impossible and we have the required contradiction.

As a corollary to (A) we see that Λ_H is incomplete and in fact that the extent of $\Lambda_H(Dt)$ contains the extent of $\Lambda_H(\Omega)$ for any ordinal formula Ω . This result, suggested to me first by the solution of question (b), may also be obtained more directly. In fact, if a number-theoretic theorem can be proved in any particular P_Ω , it can also be proved in $P_{\lambda_{mn.m(n,1,4)}}$. The formulae describing number-theoretic theorems in P do not involve more than a finite number of types, type 3 being the highest necessary. The formulae describing the number-theoretic theorems in any P_Ω will be obtained by doubling the type subscripts. Now suppose that we have a proof of a number-theoretic theorem G in P_Ω and that the types occurring in the proof are among 0, 2, 4, 6, t_1, t_2, t_3, \dots . We may suppose that they have been arranged with all the even types preceding all the odd types, the even types in order of magnitude and the type $2m-1$ preceding $2n-1$ if $\Omega(\mathbf{m}, \mathbf{n}) \text{ conv } 2$. Now let each t_r be replaced by $10+2r$ throughout the proof of G . We thus obtain a proof of G in $P_{\lambda_{mn.(n,1,4)}}$.

As with problem (a), the solution of problem (b) does not require the use of high ordinals [e.g. if we make the assumption that the extent of $\Lambda(\Omega)$ is a steadily increasing function of the ordinal represented by Ω we do not have to consider ordinals higher than $\omega+2$]. However, if we restrict what we are to call ordinal formulae in some way, we shall have corresponding modified problems (a) and (b); the solutions will presumably be essentially the same, but will involve higher ordinals. Suppose, for example, that Prod is a W.F.F. with the property that $\text{Prod}(\Omega_1, \Omega_2)$ is an ordinal formula representing $a_1 a_2$ when Ω_1, Ω_2 are ordinal formulae representing a_1, a_2 respectively, and suppose that we call a W.F.F. a 1-ordinal

formula when it is convertible to the form $\text{Sum}(\text{Prod}(\Omega, \text{Dt}), \mathbf{P})$, where Ω, \mathbf{P} are ordinal formulae of which \mathbf{P} represents a finite ordinal. We may define 1-ordinal logics, 1-completeness and 1-invariance in an obvious way, and obtain a solution of problem (b) which differs from the solution in the ordinary case in that the ordinals less than ω^2 take the place of the finite ordinals. More generally the cases that I have in mind are covered by the following theorem.

Suppose that we have a class V of formulae representing ordinals in some manner which we do not propose to specify definitely, and a subset† U of the class V such that:

(i) There is a formula Φ such that if \mathbf{T} enumerates a sequence of members of U representing an increasing sequence of ordinals, then $\Phi(\mathbf{T})$ is a member of U representing the limit of the sequence.

(ii) There is a formula \mathbf{E} such that $\mathbf{E}(m, n)$ is a member of U for each pair of positive integers m, n and, if it represents $\epsilon_{m, n}$, then $\epsilon_{m, n} < \epsilon_{m', n'}$ if either $m < m'$ or $m = m', n < n'$.

(iii) There is a formula \mathbf{G} such that, if \mathbf{A} is a member of U , then $\mathbf{G}(\mathbf{A})$ is a member of U representing a larger ordinal than does \mathbf{A} , and such that $\mathbf{G}(\mathbf{E}(m, n))$ always represents an ordinal not larger than $\epsilon_{m, n+1}$.

We define a V -ordinal logic to be a W.F.F. Λ such that $\Lambda(\mathbf{A})$ is a logic whenever \mathbf{A} belongs to V . Λ is V -invariant if the extent of $\Lambda(\mathbf{A})$ depends only on the ordinal represented by \mathbf{A} . Then it is not possible for a V -ordinal logic Λ to be V -invariant and have the property that, if \mathbf{C}_1 represents a greater ordinal than \mathbf{C}_2 (\mathbf{C}_1 and \mathbf{C}_2 both being members of U), then the extent of $\Lambda(\mathbf{C}_1)$ is greater than the extent of $\Lambda(\mathbf{C}_2)$.

We suppose the contrary. Let \mathbf{B} be a formula belonging to the extent of $\Lambda(\Phi(\lambda r. \mathbf{E}(r, 1)))$ but not to the extent of $\Lambda(\Phi(\lambda r. \mathbf{E}(r, 1)))$,

and let $\mathbf{K}' \rightarrow \lambda a. \Lambda(\mathbf{G}(\Phi(\lambda r. \text{Hg}(a, r, \mathbf{E})), \mathbf{B}))$.

Then \mathbf{K}' is a complete logic. For

$$\text{Hg}(\mathbf{A}, \mathbf{r}, \mathbf{E}) \text{ conv } \mathbf{E}(m_r, n_r).$$

† The subset U wholly supersedes V in what follows. The introduction of V serves to emphasise the fact that the set of ordinals represented by members of U may have gaps.

$E(m_r, n_r)$ is a sequence of V -ordinal formulae representing an increasing sequence of ordinals. Their limit is represented by $\Phi(\lambda r. Hg(A, r, E))$; let us see what this limit is. First suppose that A is dual: then m_r tends to infinity as r tends to infinity, and $\Phi(\lambda r. Hg(A, r, E))$ therefore represents the same ordinal as $\Phi(\lambda r. E(r, 1))$. In this case we must have

$$K'(A) \text{ conv } 2.$$

Now suppose that A is not dual: m_r is eventually equal to some constant number, a say, and $\Phi(\lambda r. Hg(A, r, E))$ represents the same ordinal as $\Phi(\lambda r. E(a, r))$, which is smaller than that represented by $\Phi(\lambda r. E(r, 1))$.

B cannot therefore belong to the extent of $\Lambda(G(\Phi(\lambda r. Hg(A, r, E))))$, and $K'(A)$ is not convertible to 2. We have proved that K' is a complete logic, which is impossible.

This theorem can no doubt be improved in many ways. However, it is sufficiently general to show that, with almost any reasonable notation for ordinals, completeness is incompatible with invariance.

We can still give a certain meaning to the classification into depths with highly restricted kinds of ordinals. Suppose that we take a particular ordinal logic A and a particular ordinal formula Ψ representing the ordinal a say (preferably a large one), and that we restrict ourselves to ordinal formulae of the form $\text{Inf}(\Psi, a)$. We then have a classification into depths, but the extents of all the logics which we so obtain are contained in the extent of a single logic.

We now attempt a problem of a rather different character, that of the completeness of Λ_P . It is to be expected that this ordinal logic is complete. I cannot at present give a proof of this, but I can give a proof that it is complete as regards a simpler type of theorem than the number-theoretic theorems, viz. those of form “ $\theta(x)$ vanishes identically”, where $\theta(x)$ is primitive recursive. The proof will have to be much abbreviated since we do not wish to go into the formal details of the system P . Also there is a certain lack of definiteness in the problem as at present stated, owing to the fact that the formulae G , E , M_P were not completely defined. Our attitude here is that it is open to the sceptical reader to give detailed definitions for these formulae and then verify that the remaining details of the proof can be filled in, using his definition. It is not asserted that these details can be filled in whatever be the definitions of G , E , M_P consistent with the properties already required of them, only that they can be filled in with the more natural definitions.

I shall prove the completeness theorem in the following form. If $\mathfrak{B}[x_0]$ is a recursion formula and if $\mathfrak{B}[0], \mathfrak{B}[f0], \dots$ are all provable in P , then there is a C-K ordinal formula \mathbf{A} such that $(x_0)\mathfrak{B}[x_0]$ is provable in the system $P^{\mathbf{A}}$ of logic obtained from P by adjoining as axioms all formulae whose G.R.'s are of the form

$$\mathbf{A} \left(\lambda mn . m \left(\varpi(2, n), \varpi(3, n) \right), K, M_P, \mathbf{r} \right)$$

(provided they represent propositions).

First let us define the formula \mathbf{A} . Suppose that \mathbf{D} is a W.F.F. with the property that $\mathbf{D}(\mathbf{n}) \text{ conv } 2$ if $\mathfrak{B}[f^{(n-1)}0]$ is provable in P , but $\mathbf{D}(\mathbf{n}) \text{ conv } 1$ if $\sim \mathfrak{B}[f^{(n-1)}0]$ is provable in P (P is being assumed consistent). Let Θ be defined by

$$\Theta \rightarrow \left\{ \lambda vu . u \left(v(v, u) \right) \right\} \left(\lambda vu . u \left(v(v, u) \right) \right),$$

and let \mathbf{Vi} be a formula with the properties

$$\mathbf{Vi}(2) \text{ conv } \lambda u . u(\text{Suc}, U),$$

$$\mathbf{Vi}(1) \text{ conv } \lambda u . u(I, \Theta(\text{Suc})).$$

The existence of such a formula is established in Kleene [1], corollary on p. 220. Now put

$$\mathbf{A}^* \rightarrow \lambda ufx . u \left(\lambda y . \mathbf{Vi} \left(\mathbf{D}(y), y, u, f, x \right) \right),$$

$$\mathbf{A} \rightarrow \text{Suc}(\mathbf{A}^*).$$

I assert that \mathbf{A}^* , \mathbf{A} are C-K ordinal formulae whenever it is true that $\mathfrak{B}[0], \mathfrak{B}[f0], \dots$ are all provable in P . For in this case \mathbf{A}^* is $\lambda ufx . u(\mathbf{R})$, where

$$\mathbf{R} \rightarrow \lambda y . \mathbf{Vi} \left(\mathbf{D}(y), y, u, f, x \right),$$

and then

$$\lambda ufx . \mathbf{R}(\mathbf{n}) \text{ conv } \lambda ufx . \mathbf{Vi} \left(\mathbf{D}(\mathbf{n}), \mathbf{n}, u, f, x \right)$$

$$\text{conv } \lambda ufx . \mathbf{Vi}(2, \mathbf{n}, u, f, x)$$

$$\text{conv } \lambda ufx . \{ \lambda n . n(\text{Suc}, U) \}(\mathbf{n}, u, f, x)$$

$\text{conv } \lambda ufx . \mathbf{n}(\text{Suc}, U, u, f, x)$, which is a C-K ordinal formula, and

$$\lambda ufx . S(\mathbf{n}, \text{Suc}, U, u, f, x) \text{ conv } \text{Suc} \left(\lambda ufx . \mathbf{n}(\text{Suc}, U, u, f, x) \right).$$

These relations hold for an arbitrary positive integer n and therefore A^* is a C-K ordinal formula [condition (9) p. 181]: it follows immediately that A is also a C-K ordinal formula. It remains to prove that $(x_0)\mathfrak{B}[x_0]$ is provable in P^A . To do this it is necessary to examine the structure of A^* in the case in which $(x_0)\mathfrak{B}[x_0]$ is false. Let us suppose that $\sim \mathfrak{B}[f^{(a-1)}0]$ is true, so that $D(a) \text{ conv } 1$, and let us consider B where

$$B \rightarrow \lambda ufx . Vi(D(a), a, u, f, x).$$

If A^* was a C-K ordinal formula, then B would be a member of its fundamental sequence; but

$$\begin{aligned} & B \text{ conv } \lambda ufx . Vi(1, a, u, f, x) \\ & \text{conv } \lambda ufx . \{ \lambda u . u(I, \Theta(\text{Suc})) \} (a, u, f, x) \\ & \text{conv } \lambda ufx . \Theta(\text{Suc}, u, f, x) \\ & \text{conv } \lambda ufx . \{ \lambda u . u(\Theta(u)) \} (\text{Suc}, u, f, x) \\ & \text{conv } \lambda ufx . \text{Suc}(\Theta(\text{Suc}), u, f, x) \\ & \text{conv } \text{Suc}(\lambda ufx . \Theta(\text{Suc}, u, f, x)) \\ & \text{conv } \text{Suc}(B). \end{aligned} \tag{9.3}$$

This, of course, implies that $B < B$ and therefore that B is no C-K ordinal formula. This, although fundamental in the possibility of proving our completeness theorem, does not form an actual step in the argument. Roughly speaking, our argument amounts to this. The relation (9.3) implies that the system P^B is inconsistent and therefore that P^{A^*} is inconsistent and indeed we can prove in P (and *a fortiori* in P^A) that $\sim (x_0)\mathfrak{B}[x_0]$ implies the inconsistency of P^{A^*} . On the other hand in P^A we can prove the consistency of P^{A^*} . The inconsistency of P^B is proved by the Gödel argument. Let us return to the details.

The axioms in P^B are those whose G.R.'s are of the form

$$B(\lambda mn . m(\varpi(2, n), \varpi(3, n)), K, M_P, r).$$

When we replace \mathbf{B} , by $\text{Suc}(\mathbf{B})$, this becomes

$$\begin{aligned} \text{Suc}(\mathbf{B}, \lambda mn.m(\varpi(2, n), \varpi(3, n)), K, M_P, \mathbf{r}) \\ \text{conv } K \left(\mathbf{B} \left(\lambda mn.m(\varpi(2, n), \varpi(3, n)), K, M_P, \mathbf{r} \right) \right) \\ \text{conv } \mathbf{B} \left(\lambda mn.m(\varpi(2, n), \varpi(3, n)), K, M_P, \mathbf{p} \right) \end{aligned}$$

if $\mathbf{r} \text{ conv } 2\mathbf{p} + 1$,

$$\text{conv } E \left(\mathbf{B} \left(\lambda mn.m(\varpi(2, n), \varpi(3, n)), K, M_P \right), \mathbf{p} \right)$$

if $\mathbf{r} \text{ conv } 2\mathbf{p}$.

When we remember the essential property of the formula E , we see that the axioms of P^B include all formulae of the form

$$(\exists x_0) \text{Proof}_{P^B}[x_0, f^{(q)}0] \supset \mathfrak{F},$$

where q is the G.R. of the formula \mathfrak{F} .

Let b be the G.R. of the formula \mathfrak{U} .

$$\sim (\exists x_0)(\exists y_0)\{\text{Proof}_{P^B}[x_0, y_0] \cdot \text{Sb}[z_0, z_0, y_0]\}. \quad (\mathfrak{U})$$

$\text{Sb}[x_0, y_0, z_0]$ is a particular recursion formula such that $\text{Sb}[f^{(l)}0, f^{(m)}0, f^{(n)}0]$ holds if and only if n is the G.R. of the result of substituting $f^{(m)}0$ for z_0 in the formula whose G.R. is l at all points where z_0 is free. Let p be the G.R. of the formula \mathfrak{C} .

$$\sim (\exists x_0)(\exists y_0)\{\text{Proof}_{P^B}[x_0, y_0] \cdot \text{Sb}[f^{(b)}0, f^{(b)}0, y_0]\}. \quad (\mathfrak{C})$$

Then we have as an axiom in P

$$(\exists x_0) \text{Proof}_{P^B}[x_0, f^{(p)}0] \supset \mathfrak{C},$$

and we can prove in P^A

$$(x_0)\{\text{Sb}[f^{(b)}0, f^{(b)}0, x_0] \equiv x_0 = f^{(p)}0\}, \quad (9.4)$$

since \mathfrak{C} is the result of substituting $f^{(b)}0$ for z_0 in \mathfrak{U} ; hence

$$\sim (\exists y_0) \text{Proof}_{P^B}[y_0, f^{(p)}0] \quad (9.5)$$

is provable in P . Using (9.4) again, we see that \mathfrak{C} can be proved in P^B . But, if we can prove \mathfrak{C} in P^B , then we can prove its provability in P^B , the

proof being in P ; i.e. we can prove

$$(\exists x_0) \text{Proof}_{P^B} [x_0, f^{(p)} 0]$$

in P (since p is the G.R. of \mathfrak{C}). But this contradicts (9.5), so that, if

$$\sim \mathfrak{B} [f^{(a-1)} 0]$$

is true, we can prove a contradiction in P^B or in P^{A^*} . Now I assert that the whole argument up to this point can be carried through formally in the system P , in fact, that, if c is the G.R. of $\sim (0 = 0)$, then

$$\sim (x_0) \mathfrak{B} [x_0] \supset (\exists v_0) \text{Proof}_{P^{A^*}} [v_0, f^{(c)} 0] \quad (9.6)$$

is provable in P . I shall not attempt to give any more detailed proof of this assertion.

The formula

$$(\exists x_0) \text{Proof}_{P^{A^*}} [x_0, f^{(c)} 0] \supset \sim (0 = 0) \quad (9.7)$$

is an axiom in P^A . Combining (9.6), (9.7) we obtain $(x_0) \mathfrak{B} [x_0]$ in P^A .

This completeness theorem as usual is of no value. Although it shows, for instance, that it is possible to prove Fermat's last theorem with Λ_P (if it is true) yet the truth of the theorem would really be assumed by taking a certain formula as an ordinal formula.

That Λ_P is not invariant may be proved easily by our general theorem; alternatively it follows from the fact that, in proving our partial completeness theorem, we never used ordinals higher than $\omega+1$. This fact can also be used to prove that Λ_P is not C-K invariant up to $\omega+2$.

10. *The continuum hypothesis. A digression.*

The methods of §9 may be applied to problems which are constructive analogues of the continuum hypothesis problem. The continuum hypothesis asserts that $2^{\aleph_0} = \aleph_1$, in other words that, if ω_1 is the smallest ordinal α greater than ω such that a series with order type α cannot be put into one-one correspondence with the positive integers, then the ordinals less than ω_1 can be put into one-one correspondence with the subsets of the positive integers. To obtain a constructive analogue of this proposition we may replace the ordinals less than ω_1 either by the ordinal formulae, or by the ordinals represented by them; we may replace the subsets of the positive integers either by the computable sequences of figures 0, 1, or by the description numbers of the machines which compute these sequences. In the manner in which the correspondence is to be set up there is also more than one possibility. Thus, even when we use only

one kind of ordinal formula, there is still great ambiguity concerning what the constructive analogue of the continuum hypothesis should be. I shall prove a single result in this connection†. A number of others may be proved in the same way.

We ask “Is it possible to find a computable function of ordinal formulae determining a one-one correspondence between the ordinals represented by ordinal formulae and the computable sequences of figures 0, 1?” More accurately, “Is there a formula F such that if Ω is an ordinal formula and n a positive integer then $F(\Omega, n)$ is convertible to 1 or to 2, and such that $F(\Omega, n) \text{ conv } F(\Omega', n)$ for each positive integer n , if and only if Ω and Ω' represent the same ordinal?” The answer is “No”, as will be seen to be a consequence of the following argument: there is no formula F such that $F(\Omega)$ enumerates one sequence of integers (each being 1 or 2) when Ω represents ω and enumerates another sequence when Ω represents 0. If there is such an F , then there is an a such that $F(\Omega, a) \text{ conv } (Dt, a)$ if Ω represents ω but $F(\Omega, a)$ and $F(Dt, a)$ are convertible to different integers (1 or 2) if Ω represents 0. To obtain a contradiction from this we introduce a W.F.F. Gm not unlike Mg . If the machine M whose D.N. is n has printed 0 by the time the m -th complete configuration is reached then

$$Gm(n, m) \text{ conv } \lambda mn . m(n, I, 4);$$

otherwise $Gm(n, m) \text{ conv } \lambda pq . A1(4(P, 2p+2q), 3, 4)$. Now consider $F(Dt, a)$ and $F(\text{Lim}(Gm(n)), a)$. If M never prints 0, $\text{Lim}(Gm(n))$ represents the ordinal ω . Otherwise it represents 0. Consequently these two formulae are convertible to one another if and only if M never prints 0. This gives us a means of determining about any machine whether it ever prints 0, which is impossible.

Results of this kind have of course no real relevance for the classical continuum hypothesis.

11. The purpose of ordinal logics.

Mathematical reasoning may be regarded rather schematically as the exercise of a combination of two faculties‡, which we may call *intuition* and *ingenuity*. The activity of the intuition consists in making spontaneous judgments which are not the result of conscious trains

† A suggestion to consider this problem came to me indirectly from F. Bernstein. A related problem was suggested by P. Bernays.

‡ We are leaving out of account that most important faculty which distinguishes topics of interest from others; in fact, we are regarding the function of the mathematician as simply to determine the truth or falsity of propositions.

of reasoning. These judgments are often but by no means invariably correct (leaving aside the question what is meant by "correct"). Often it is possible to find some other way of verifying the correctness of an intuitive judgment. We may, for instance, judge that all positive integers are uniquely factorizable into primes; a detailed mathematical argument leads to the same result. This argument will also involve intuitive judgments, but they will be less open to criticism than the original judgment about factorization. I shall not attempt to explain this idea of "intuition" any more explicitly.

The exercise of ingenuity in mathematics consists in aiding the 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.

The parts played by these two faculties differ of course from occasion to occasion, and from mathematician to mathematician. This arbitrariness can be removed by the introduction of a formal logic. The necessity for using the intuition is then greatly reduced by setting down formal rules for carrying out inferences which are always intuitively valid. When working with a formal logic, the idea of ingenuity takes a more definite shape. In general a formal logic, will be framed so as to admit a considerable variety of possible steps in any stage in a proof. Ingenuity will then determine which steps are the more profitable for the purpose of proving a particular proposition. In pre-Gödel times it was thought by some that it would probably be possible to carry this programme to such a point that all the intuitive judgments of mathematics could be replaced by a finite number of these rules. The necessity for intuition would then be entirely eliminated.

In our discussions, however, we have gone to the opposite extreme and eliminated not intuition but ingenuity, and this in spite of the fact that our aim has been in much the same direction. We have been trying to see how far it is possible to eliminate intuition, and leave only ingenuity. We do not mind how much ingenuity is required, and therefore assume it to be available in unlimited supply. In our metamathematical discussions we actually express this assumption rather differently. We are always able to obtain from the rules of a formal logic a method of enumerating the propositions proved by its means. We then imagine that all proofs take the form of a search through this enumeration for the theorem for which a proof is desired. In this way ingenuity is replaced by patience. In these heuristic discussions, however, it is better not to make this reduction.

In consequence of the impossibility of finding a formal logic which wholly eliminates the necessity of using intuition, we naturally turn to "non-constructive" systems of logic with which not all the steps in a proof are mechanical, some being intuitive. An example of a non-constructive logic is afforded by any ordinal logic. When we have an ordinal logic, we are in a position to prove number-theoretic theorems by the intuitive steps of recognizing formulae as ordinal formulae, and the mechanical steps of carrying out conversions. What properties do we desire a non-constructive logic to have if we are to make use of it for the expression of mathematical proofs? We want it to show quite clearly when a step makes use of intuition, and when it is purely formal. The strain put on the intuition should be a minimum. Most important of all, it must be beyond all reasonable doubt that the logic leads to correct results whenever the intuitive steps are correct[†]. It is also desirable that the logic shall be adequate for the expression of number-theoretic theorems, in order that it may be used in metamathematical discussions (cf. § 5).

Of the particular ordinal logics that we have discussed, Λ_H and Λ_P certainly will not satisfy us. In the case of Λ_H we are in no better position than with a constructive logic. In the case of Λ_P (and for that matter also Λ_H) we are by no means certain that we shall never obtain any but true results, because we do not know whether all the number-theoretic theorems provable in the system P are true. To take Λ_P as a fundamental non-constructive logic for metamathematical arguments would be most unsound. There remains the system of Church which is free from these objections. It is probably complete (although this would not necessarily mean much) and it is beyond reasonable doubt that it always leads to correct results[‡]. In the next section I propose to describe another ordinal logic, of a very different type, which is suggested by the work of Gentzen and which should also be adequate for the formalization of number-theoretic theorems. In particular it should be suitable for proofs of metamathematical theorems (cf. § 5).

[†] This requirement is very vague. It is not of course intended that the criterion of the correctness of the intuitive steps be the correctness of the final result. The meaning becomes clearer if each intuitive step is regarded as a judgment that a particular proposition is true. In the case of an ordinal logic it is always a judgment that a formula is an ordinal formula, and this is equivalent to judging that a number-theoretic proposition is true. In this case then the requirement is that the reputed ordinal logic *is* an ordinal logic.

[‡] This ordinal logic arises from a certain system C_0 in essentially the same way as Λ_P arose from P . By an argument similar to one occurring in § 8 we can show that the ordinal logic leads to correct results if and only if C_0 is valid; the validity of C_0 is proved in Church [1], making use of the results of Church and Rosser [1].

12. *Gentzen type ordinal logics.*

In proving the consistency of a certain system of formal logic Gentzen (Gentzen [1]) has made use of the principle of transfinite induction for ordinals less than ϵ_0 , and has suggested that it is to be expected that transfinite induction carried sufficiently far would suffice to solve all problems of consistency. Another suggestion of basing systems of logic on transfinite induction has been made by Zermelo (Zermelo [1]). In this section I propose to show how this method of proof may be put into the form of a formal (non-constructive) logic, and afterwards to obtain from it an ordinal logic.

We can express the Gentzen method of proof formally in this way. Let us take the system P and adjoin to it an axiom \mathfrak{U}_Ω with the intuitive meaning that the W.F.F. Ω is an ordinal formula, whenever we feel certain that Ω is an ordinal formula. This is a non-constructive system of logic which may easily be put into the form of an ordinal logic. By the method of § 6 we make correspond to the system of logic consisting of P with the axiom \mathfrak{U}_Ω adjoined a logic formula L_Ω : L_Ω is an effectively calculable function of Ω , and there is therefore a formula Λ_g^{-1} such that $\Lambda_g^{-1}(\Omega) \text{ conv } L_\Omega$ for each formula Ω . Λ_g^{-1} is certainly not an ordinal logic unless P is valid, and therefore consistent. This formalization of Gentzen's idea would therefore not be applicable for the problem with which Gentzen himself was concerned, for he was proving the consistency of a system weaker than P . However, there are other ways in which the Gentzen method of proof can be formalized. I shall explain one, beginning by describing a certain logical calculus.

The symbols of the calculus are $f, x, ^1, \mathfrak{j}, 0, S, R, \Gamma, \Delta, E, |, \Theta, !, (,), =$, and the comma “,”. For clarity we shall use various sizes of brackets $(,)$ in the following. We use capital German letters to stand for variable or undetermined sequences of these symbols.

It is to be understood that the relations that we are about to define hold only when compelled to do so by the conditions that we lay down. The conditions should be taken together as a simultaneous inductive definition of all the relations involved.

Suffixes.

${}_1$ is a suffix. If \mathfrak{S} is a suffix then \mathfrak{S}_1 is a suffix.

Indices.

1 is an index. If \mathfrak{I} is an index then \mathfrak{I}^1 is an index.

Numerical variables.

■ If \mathfrak{S} is a suffix then $x\mathfrak{S}$ is a numerical variable.

Functional variables.

If \mathfrak{G} is a suffix and \mathfrak{I} is an index, then $f\mathfrak{G}\mathfrak{I}$ is a functional variable of index \mathfrak{I} .

Arguments.

$(,)$ is an argument of index 1 . If (\mathfrak{A}) is an argument of index \mathfrak{I} and \mathfrak{T} is a term, then $(\mathfrak{A}\mathfrak{T},)$ is an argument of index \mathfrak{I}^1 .

Numerals.

0 is a numeral.

If \mathfrak{N} is a numeral, then $S(, \mathfrak{N},)$ is a numeral.

In metamathematical statements we shall denote the numeral in which S occurs r times by $S^{(r)}(, 0,)$.

Expressions of a given index.

A functional variable of index \mathfrak{I} is an expression of index \mathfrak{I} .

R, S are expressions of index $^{111}, ^{11}$ respectively.

If \mathfrak{N} is a numeral, then it is also an expression of index 1 .

Suppose that \mathfrak{G} is an expression of index \mathfrak{I} , \mathfrak{H} one of index \mathfrak{I}^1 and \mathfrak{K} one of index \mathfrak{I}^{111} ; then $(\Gamma\mathfrak{G})$ and $(\Delta\mathfrak{G})$ are expressions of index \mathfrak{I} , while $(E\mathfrak{G})$ and $(\mathfrak{G}|\mathfrak{H})$ and $(\mathfrak{G}\odot\mathfrak{K})$ and $(\mathfrak{G}!|\mathfrak{H}!|\mathfrak{K})$ are expressions of index \mathfrak{I}^1 .

Function constants.

An expression of index \mathfrak{I} in which no functional variable occurs is a function constant of index \mathfrak{I} . If in addition R does not occur, the expression is called a *primitive function constant*.

Terms.

0 is a term.

Every numerical variable is a term.

If \mathfrak{G} is an expression of index \mathfrak{I} and (\mathfrak{A}) is an argument of index \mathfrak{I} , then $\mathfrak{G}(\mathfrak{A})$ is a term.

Equations.

If \mathfrak{T} and \mathfrak{T}' are terms, then $\mathfrak{T} = \mathfrak{T}'$ is an equation.

Provable equations.

We define what is meant by the provable equations relative to a given set of equations as axioms.

(a) The provable equations include all the axioms. The axioms are of the form of equations in which the symbols Γ , Δ , E , $|$, \odot , $!$ do not appear.

(b) If \mathfrak{G} is an expression of index \mathfrak{I}^{11} and (\mathfrak{A}) is an argument of index \mathfrak{I} , then

$$(\Gamma\mathfrak{G})(\mathfrak{A}x_1, x_{11},) = \mathfrak{G}(\mathfrak{A}x_{11}, x_1,)$$

is a provable equation.

(c) If \mathfrak{G} is an expression of index \mathfrak{I}^1 , and (\mathfrak{A}) is an argument of index \mathfrak{I} , then

$$(\Delta\mathfrak{G})(\mathfrak{A}x_1,) = \mathfrak{G}(, x_1 \mathfrak{A})$$

is a provable equation.

(d) If \mathfrak{G} is an expression of index \mathfrak{I} , and (\mathfrak{A}) is an argument of index \mathfrak{I} , then

$$(E\mathfrak{G})(\mathfrak{A}x_1,) = \mathfrak{G}(\mathfrak{A})$$

is a provable equation.

(e) If \mathfrak{G} is an expression of index \mathfrak{I} and \mathfrak{H} is one of index \mathfrak{I}^1 , and (\mathfrak{A}) is an argument of index \mathfrak{I} , then

$$(\mathfrak{G}|\mathfrak{H})(\mathfrak{A}) = \mathfrak{H}(\mathfrak{A}\mathfrak{G}(\mathfrak{A}),)$$

is a provable equation.

(f) If \mathfrak{N} is an expression of index 1 , then $\mathfrak{N}(,) = \mathfrak{N}$ is a provable equation.

(g) If \mathfrak{G} is an expression of index \mathfrak{I} and \mathfrak{K} one of index \mathfrak{I}^{111} , and (\mathfrak{A}) an argument of index \mathfrak{I}^1 , then

$$(\mathfrak{G} \odot \mathfrak{K})(\mathfrak{A}0,) = \mathfrak{G}(\mathfrak{A})$$

and $(\mathfrak{G} \odot \mathfrak{K})(\mathfrak{A}S(, x_1,),) = \mathfrak{K}(\mathfrak{A}x_1, S(, x_1,), (\mathfrak{G} \odot \mathfrak{K})(\mathfrak{A}x_1,),)$

are provable equations. If in addition \mathfrak{H} is an expression of index \mathfrak{I}^1 and

$$R\left(, \mathfrak{G}(\mathfrak{A}S(, x_1,),), x_1, \right) = 0$$

is provable, then

$$(\mathfrak{G}! \mathfrak{K}! \mathfrak{H})(\mathfrak{A}0,) = \mathfrak{G}(\mathfrak{A})$$

and

$$(\mathfrak{G}! \mathfrak{K}! \mathfrak{H})(\mathfrak{A}S(, x_1,),)$$

$$= \mathfrak{K}\left(\left(\mathfrak{A}\mathfrak{H}(\mathfrak{A}S(, x_1,),), S(, x_1,), (\mathfrak{G}! \mathfrak{K}! \mathfrak{H})(\mathfrak{A}\mathfrak{H}(\mathfrak{A}S(, x_1,),),),\right),\right)$$

are provable.

(h) If $\mathfrak{X} = \mathfrak{X}'$ and $\mathfrak{U} = \mathfrak{U}'$ are provable, where \mathfrak{X} , \mathfrak{X}' , \mathfrak{U} and \mathfrak{U}' are terms, then $\mathfrak{U}' = \mathfrak{U}$ and the result of substituting \mathfrak{U}' for \mathfrak{U} at any particular occurrence in $\mathfrak{X} = \mathfrak{X}'$ are provable equations.

(i) The result of substituting any term for a particular numerical variable throughout a provable equation is provable.

(j) Suppose that \mathfrak{G} , \mathfrak{G}' are expressions of index \mathfrak{I}^1 , that (\mathfrak{U}) is an argument of index \mathfrak{I} not containing the numerical variable \mathfrak{X} and that $\mathfrak{G}(\mathfrak{U}0,) = \mathfrak{G}'(\mathfrak{U}0,)$ is provable. Also suppose that, if we add

$$\mathfrak{G}(\mathfrak{U}\mathfrak{X},) = \mathfrak{G}'(\mathfrak{U}\mathfrak{X},)$$

to the axioms and restrict (i) so that it can never be applied to the numerical variable \mathfrak{X} , then

$$\mathfrak{G}(\mathfrak{U}S(\mathfrak{X},),) = \mathfrak{G}'(\mathfrak{U}S(\mathfrak{X},),)$$

becomes a provable equation; in the hypothetical proof of this equation this rule (j) itself may be used provided that a different variable is chosen to take the part of \mathfrak{X} .

Under these conditions $\mathfrak{G}(\mathfrak{U}\mathfrak{X},) = \mathfrak{G}'(\mathfrak{U}\mathfrak{X},)$ is a provable equation.

(k) Suppose that \mathfrak{G} , \mathfrak{G}' , \mathfrak{H} are expressions of index \mathfrak{I}^1 , that (\mathfrak{U}) is an argument of index \mathfrak{I} not containing the numerical variable \mathfrak{X} and that

$$\mathfrak{G}(\mathfrak{U}0,) = \mathfrak{G}'(\mathfrak{U}0,) \quad \text{and} \quad R(\mathfrak{H}(\mathfrak{U}S(\mathfrak{X},),), S(\mathfrak{X},),) = 0$$

are provable equations. Suppose also that, if we add

$$\mathfrak{G}(\mathfrak{U}\mathfrak{H}(\mathfrak{U}S(\mathfrak{X},),)) = \mathfrak{G}'(\mathfrak{U}\mathfrak{H}(\mathfrak{U}S(\mathfrak{X},),))$$

to the axioms, and again restrict (i) so that it does not apply to \mathfrak{X} , then

$$\mathfrak{G}(\mathfrak{U}\mathfrak{X},) = \mathfrak{G}'(\mathfrak{U}\mathfrak{X},) \tag{12.1}$$

becomes a provable equation; in the hypothetical proof of (12.1) the rule (b) may be used if a different variable takes the part of \mathfrak{X} .

Under these conditions (12.1) is a provable equation.

We have now completed the definition of a provable equation relative to a given set of axioms. Next we shall show how to obtain an ordinal logic from this calculus. The first step is to set up a correspondence between some of the equations and number-theoretic theorems, in order to show how they can be interpreted as number-theoretic theorems.

Let \mathfrak{G} be a primitive function constant of index ¹¹¹. \mathfrak{G} describes a certain primitive recursive function $\phi(m, n)$, determined by the condition that, for all natural numbers m, n , the equation

$$\mathfrak{G} \left(, S^{(m)}(, 0,), S^{(n)}(, 0,), \right) = S^{(\phi(m, n))}(, 0,)$$

is provable without using the axioms (a). Suppose also that \mathfrak{H} is an expression of index \mathfrak{J} . Then to the equation

$$\mathfrak{G} \left(, x_1, \mathfrak{H}(, x_1,), \right) = 0$$

we make correspond the number-theoretic theorem which asserts that for each natural number m there is a natural number n such that $\phi(m, n) = 0$. (The circumstance that there is more than one equation to represent each number-theoretic theorem could be avoided by a trivial but inconvenient modification of the calculus.)

Now let us suppose that some definite method is chosen for describing the sets of axioms by means of positive integers, the null set of axioms being described by the integer 1. By an argument used in § 6 there is a W.F.F. Σ such that, if r is the integer describing a set A of axioms, then $\Sigma(r)$ is a logic formula enabling us to prove just those number-theoretic theorems which are associated with equations provable with the above described calculus, the axioms being those described by the number r .

I explain two ways in which the construction of the ordinal logic may be completed.

In the first method we make use of the theory of general recursive functions (Kleene [2]). Let us consider all equations of the form

$$R \left(, S^{(m)}(, 0,), S^{(n)}(, 0,), \right) = S^{(p)}(, 0,) \quad (12.2)$$

which are obtainable from the axioms by the use of rules (h), (i). It is a consequence of the theorem of equivalence of λ -definable and general recursive functions (Kleene [3]) that, if $r(m, n)$ is any λ -definable function of two variables, then we can choose the axioms so that (12.2) with $p = r(m, n)$ is obtainable in this way for each pair of natural numbers m, n , and no equation of the form

$$S^{(m)}(, 0,) = S^{(n)}(, 0,) \quad (m \neq n) \quad (12.3)$$

is obtainable. In particular, this is the case if $r(m, n)$ is defined by the condition that

$$\Omega(\mathbf{m}, \mathbf{n}) \text{ conv } S(\mathbf{p}) \text{ implies } p = r(m, n),$$

$$r(0, n) = 1, \text{ all } n > 0, \quad \underline{r(0, 0) = 2},$$

where Ω is an ordinal formula. There is a method for obtaining the axioms given the ordinal formula, and consequently a formula Rec such that, for any ordinal formula Ω , $\text{Rec}(\Omega) \text{ conv } m$, where m is the integer describing the set of axioms corresponding to Ω . Then the formula

$$\Lambda_G^2 \rightarrow \lambda w. \Sigma(\text{Rec}(w))$$

is an ordinal logic. Let us leave the proof of this aside for the present.

Our second ordinal logic is to be constructed by a method not unlike the one which we used in constructing Λ_P . We begin by assigning ordinal formulae to all sets of axioms satisfying certain conditions. For this purpose we again consider that part of the calculus which is obtained by restricting "expressions" to be functional variables or R or S and restricting the meaning of "term" accordingly; the new provable equations are given by conditions (a), (h), (i), together with an extra condition (l).

(l) The equation

$$R(, 0, S(, x_1,)) = 0$$

is provable.

We could design a machine which would obtain all equations of the form (12.2), with $m \neq n$, provable in this sense, and all of the form (12.3), except that it would cease to obtain any more equations when it had once obtained one of the latter "contradictory" equations. From the description of the machine we obtain a formula Ω such that

$$\Omega(m, n) \text{ conv } 2 \quad \text{if} \quad R(, S^{(m-1)}(, 0,), S^{(n-1)}(, 0,)) = 0$$

is obtained by the machine,

$$\Omega(m, n) \text{ conv } 1 \quad \text{if} \quad R(, S^{(n-1)}(, 0,), S^{(m-1)}(, 0,)) = 0$$

is obtained by the machine, and

$$\Omega(m, m) \text{ conv } 3 \text{ always.}$$

The formula Ω is an effectively calculable function of the set of axioms, and therefore also of m : consequently there is a formula M such that $M(m) \text{ conv } \Omega$ when m describes the set of axioms. Now let Cm be a formula such that, if b is the G.R. of a formula $M(m)$, then $Cm(b) \text{ conv } m$, but otherwise $Cm(b) \text{ conv } 1$. Let

$$\Lambda_G^3 \rightarrow \lambda wa. \Gamma(\lambda n. \Sigma(Cm(Tn(w, n))), a).$$

Then $\Lambda_G^3(\Omega, A)$ conv 2 if and only if Ω conv $M(m)$, where m describes a set of axioms which, taken with our calculus, suffices to prove the equation which is, roughly speaking, equivalent to "A is dual". To prove that Λ_G^3 is an ordinal logic, it is sufficient to prove that the calculus with the axioms described by m proves only true number-theoretic theorems when Ω is an ordinal formula. This condition on m may also be expressed in this way. Let us put $m \ll n$ if we can prove $R(S^{(m)}(0, 0), S^{(n)}(0, 0)) = 0$ with (a), (h), (i), (l): the condition is that $m \ll n$ is a well-ordering of the natural numbers and that no contradictory equation (12.3) is provable with the same rules (a), (h), (i), (l). Let us say that such a set of axioms is *admissible*. Λ_G^3 is an ordinal logic if the calculus leads to none but true number-theoretic theorems when an admissible set of axioms is used.

In the case of Λ_G^2 , $\text{Rec}(\Omega)$ describes an admissible set of axioms whenever Ω is an ordinal formula. Λ_G^2 therefore is an ordinal logic if the calculus leads to correct results when admissible axioms are used.

To prove that admissible axioms have the required property, I do not attempt to do more than show how interpretations can be given to the equations of the calculus so that the rules of inference (a)–(k) become intuitively valid methods of deduction, and so that the interpretation agrees with our convention regarding number-theoretic theorems.

Each expression is the name of a function, which may be only partially defined. The expression S corresponds simply to the successor function. If \mathfrak{G} is either R or a functional variable and has $p+1$ symbols in its index, then it corresponds to a function g of p natural numbers defined as follows. If

$$\mathfrak{G}(S^{(r_1)}(0, 0), S^{(r_2)}(0, 0), \dots, S^{(r_p)}(0, 0)) = S^{(l)}(0, 0)$$

is provable by the use of (a), (h), (i), (l) only, then $g(r_1, r_2, \dots, r_p)$ has the value p . It may not be defined for all arguments, but its value is always unique, for otherwise we could prove a "contradictory" equation and $M(m)$ would then not be an ordinal formula. The functions corresponding to the other expressions are essentially defined by (b)–(f). For example, if g is the function corresponding to \mathfrak{G} and g' that corresponding to $(\Gamma\mathfrak{G})$, then

$$g'(r_1, r_2, \dots, r_p, l, m) = g(r_1, r_2, \dots, r_p, m, l).$$

The values of the functions are clearly unique (when defined at all) if given by one of (b)–(e). The case (f) is less obvious since the function defined appears also in the definiens. I do not treat the case of $(\mathfrak{G}\Theta\mathfrak{K})$, since this is the well-known definition by primitive recursion, but I shall show that the values of the function corresponding to $(\mathfrak{G}\!\!\! \cdot \mathfrak{K}\!\!\! \cdot \mathfrak{h})$ are unique. Without loss of generality we may suppose that (\mathfrak{A}) in (f) is of index ¹. We have

then to show that, if $h(m)$ is the function corresponding to \mathfrak{h} and $r(m, n)$ that corresponding to R , and $k(u, v, w)$ is a given function and a a given natural number, then the equations

$$l(0) = a, \quad (a)$$

$$l(m+1) = k\left(h(m+1), m+1, l\left(h(m+1)\right)\right) \quad (b)$$

do not ever assign two different values for the function $l(m)$. Consider those values of r for which we obtain more than one value of $l(r)$, and suppose that there is at least one such. Clearly 0 is not one, for $l(0)$ can be defined only by (a). Since the relation \ll is a well ordering, there is an integer r_0 such that $r_0 > 0$, $l(r_0)$ is not unique, and if $s \neq r_0$ and $l(s)$ is not unique then $r_0 \ll s$. We may put $s = h(r_0)$, for, if $l\left(h(r_0)\right)$ were unique, then $l(r_0)$, defined by (b), would be unique. But $r\left(h(r_0), r_0\right) = 0$ i.e. $s \ll r_0$. There is, therefore, no integer r for which we obtain more than one value for the function $l(r)$.

Our interpretation of expressions as functions gives us an immediate interpretation for equations with no numerical variables. In general we interpret an equation with numerical variables as the (infinite) conjunction of all equations obtainable by replacing the variables by numerals. With this interpretation (h), (i) are seen to be valid methods of proof. In (j) the provability of

$$\mathfrak{G}\left(\mathfrak{A}S(x_1),\right) = \mathfrak{G}'\left(\mathfrak{A}S(x_1),\right)$$

when $\mathfrak{G}(\mathfrak{A}x_1) = \mathfrak{G}'(\mathfrak{A}x_1)$ is assumed to be interpreted as meaning that the implication between these equations holds for all substitutions of numerals for x_1 . To justify this, one should satisfy oneself that these implications always hold when the hypothetical proof can be carried out. The rule of procedure (j) is now seen to be simply mathematical induction. The rule (k) is a form of transfinite induction. In proving the validity of (k) we may again suppose (A) is of index 1. Let $r(m, n)$, $g(m)$, $g_1(m)$, $h(n)$ be the functions corresponding respectively to R , \mathfrak{G} , \mathfrak{G}' , \mathfrak{h} . We shall prove that, if $g(0) = g'(0)$ and $r\left(h(n), n\right) = 0$ for each positive integer n and if $g(n+1) = g'(n+1)$ whenever $g\left(h(n+1)\right) = g'\left(h(n+1)\right)$, then $g(n) = g'(n)$ for each natural number n . We consider the class of natural numbers for which $g(n) = g'(n)$ is not true. If the class is not void it has a positive member n_0 which precedes all other members in the well ordering \ll . But $h(n_0)$ is another member of the class, for otherwise we should have

$$g\left(h(n_0)\right) = g'\left(h(n_0)\right)$$

and therefore $g(n_0) = g'(n_0)$, i.e. n_0 would not be in the class. This implies $n_0 \ll h(n_0)$ contrary to $r(h(n_0), n_0) = 0$. The class is therefore void.

It should be noticed that we do not really need to make use of the fact that Ω is an ordinal formula. It suffices that Ω should satisfy conditions (a)–(e) (p. 179) for ordinal formulae, and in place of (f) satisfy (f').

(f') There is no formula T such that $T(n)$ is convertible to a formula representing a positive integer for each positive integer n , and such that $\Omega(T(n), n) \text{ conv } 2$, for each positive integer n for which $\Omega(n, n) \text{ conv } 3$.

The problem whether a formula satisfies conditions (a)–(e), (f') is number-theoretic. If we use formulae satisfying these conditions instead of ordinal formulae with Λ_G^2 or Λ_G^3 , we have a non-constructive logic with certain advantages over ordinal logics. The intuitive judgments that must be made are all judgments of the truth of number theoretic theorems. We have seen in § 9 that the connection of ordinal logics with the classical theory of ordinals is quite superficial. There seem to be good reasons, therefore, for giving attention to ordinal formulae in this modified sense.

The ordinal logic Λ_G^3 appears to be adequate for most purposes. It should, for instance, be possible to carry out Gentzen's proof of consistency of number theory, or the proof of the uniqueness of the normal form of a well-formed formula (Church and Rosser [1]) with our calculus and a fairly simple set of axioms. How far this is the case can, of course, only be determined by experiment.

One would prefer a non-constructive system of logic based on transfinite induction rather simpler than the system which we have described. In particular, it would seem that it should be possible to eliminate the necessity of stating explicitly the validity of definitions by primitive recursions, since this principle itself can be shown to be valid by transfinite induction. It is possible to make such modifications in the system, even in such a way that the resulting system is still complete, but no real advantage is gained by doing so. The effect is always, so far as I know, to restrict the class of formulae provable with a given set of axioms, so that we obtain no theorems but trivial restatements of the axioms. We have therefore to compromise between simplicity and comprehensiveness.

Index of definitions.

No attempt is being made to list heavy type formulae since their meanings are not always constant throughout the paper. Abbreviations

for definite well-formed formulae are listed alphabetically.

				Page						Page
Ai	199	Prod	207
Al	188	Q	176
Bd	188	Rec	222
Ck	192	Rt	205
Cm	222	S	164
Comp	201	Sum...	188
Dt	167	Sq	206
E	196	Tn	192
form	166	Ug	188
G	196	V	177
Gm	214	Vi	210
Gr	166	W	175
H	183, 186	W'	176
H_1	186	X	175
Hf	188	Z	204
Hg	206						
I	163	Γ	177
Inf	188	δ	162
Jh	204	Θ	210
K	196	Λ_G^1	217
Lim	188	Λ_G^2	222
Ls	187	Λ_G^3	222
M	222	Λ_H	198
M_P	196	Λ_P	196
Mg	205	ϖ	167
Nm	177	Σ	221
Od	201	1, 2, 3,	164
P	188	\wp	170

(The following refer to §§ 1-10 only.)

All-inclusive (logic formula)	199
Axiomatic (class or property)...	167
Circle-free	(Turing [1], 233)	
Computable function	166
Completeness, of class of logics	176
of logic	177
of ordinal logic	198
Convertible	163

Description number (D.N.)	(Turing [1], 240)
Dual (W.F.F.)	170
Effectively calculable function	166
Enumerate (to)	165
Formally definable function	165
General recursive function	166
Gödel representation (G.R.)	165, 166
Immediately convertible	162
Invariance (of ordinal logics)	200
					(see also 202, 203)	
Limit system	190
Logic formula, Logic	174
Normal form	162, 165
Number-theoretic (theorem or problem)	168
Oracle	172
Ordinal	178
Ordinal formula	179, 180
C-K ordinal formula	181
Ordinal logic	189
Primitive recursive (function or relation)	168, 193
Recursion formula	193
Representation of ordinals, by ordinal formulae	179
			by C-K ordinal formulae	182
Standardized logic	175
Type	197
Validity of system	189
Well-formed formula (W.F.F.)	162
Well ordered series	178

Miscellaneous (in order of appearance).

Bibliography.

- Alonzo Church, [1]. "A proof of freedom from contradiction", *Proc. Nat. Acad. Sci.*, 21 (1935), 275–281.
- , [2]. *Mathematical logic*, Lectures at Princeton University (1935–6), mimeographed, 113 pp.
- , [3]. "An unsolvable problem of elementary number theory", *American J. of Math.*, 58 (1936), 345–363.
- , [4]. "The constructive second number class", *Bull. American Math. Soc.*, 44 (1938), 224–238.
- G. Gentzen, [1]. "Die Widerspruchsfreiheit der reinen Zahlentheorie", *Math. Annalen*, 112 (1936), 493–565.
- K. Gödel, [1]. "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte für Math. und Phys.*, 38 (1931), 173–189.
- , [2]. *On undecidable propositions of formal mathematical systems*, Lectures at the Institute for Advanced Study, Princeton, N.J., 1934, mimeographed, 30 pp.
- D. Hilbert, [1]. "Über das Unendliche", *Math. Annalen*, 95 (1926), 161–190.
- S. C. Kleene, [1]. "A theory of positive integers in formal logic", *American J. of Math.*, 57 (1935), 153–173 and 219–244.
- , [2]. "General recursive functions of natural numbers", *Math. Annalen*, 112 (1935–6), 727–742.
- , [3]. " λ -definability and recursiveness", *Duke Math. Jour.*, 2 (1936), 340–353.
- E. L. Post, [1]. "Finite combinatory processes—formulation 1", *Journal Symbolic Logic*, 1 (1936), 103–105.
- J. B. Rosser, [1]. "Gödel theorems for non-constructive logics", *Journal Symbolic Logic*, 2 (1937), 129–137.
- A. Tarski, [1]. "Der Wahrheitsbegriff in den formalisierten Sprachen", *Studia Philosophica*, 1 (1936), 261–405 (translation from the original paper in Polish dated 1933).
- A. M. Turing, [1]. "On computable numbers, with an application to the Entscheidungsproblem", *Proc. London Math. Soc.* (2), 42 (1937), 230–265. A correction to this paper has appeared in the same periodical, 43 (1937), 544–546.
- , [2]. "Computability and λ -definability", *Journal Symbolic Logic*, 2 (1937), 153–163.
- E. Zermelo, [1]. "Grundlagen einer allgemeiner Theorie der mathematischen Satzsysteme, I", *Fund. Math.*, 25 (1935), 136–146.
- Alonzo Church and S. C. Kleene, [1]. "Formal definitions in the theory of ordinal numbers", *Fund. Math.*, 28 (1936), 11–21.
- Alonzo Church and J. B. Rosser, [1]. "Some properties of conversion", *Trans. American Math. Soc.*, 39 (1936), 472–482.
- D. Hilbert and W. Ackermann, [1]. *Grundzüge der theoretischen Logik* (2nd edition revised, Berlin, 1938), 130 pp.
- A. N. Whitehead and Bertrand Russell, [1]. *Principia Mathematica* (2nd edition, Cambridge, 1925–1927), 3 vols.

King's College,
Cambridge.

Part II

Type Theory

This Page Intentionally Left Blank

General Introduction to Turing's work on Type Theory

Turing may have learnt something about the theory of types from Newman, from Hilbert and Ackermann [1928], where there is an account of the ramified theory, and from Gödel [1931], where there is a formulation of the simple theory of types. But it was Church's lectures (1937–8) which really aroused his interest. In a letter to Newman dated April 21st (it must be 1940) he writes:

Church tells me he is going to publish his form of Principia involving the use of λ and simple theory of types. I am very glad of it, as the system make things much clearer than any other system I know and is not too cumbrous to be used.

1. Church's System

(1) σ and ι are (basic) type symbols; type σ is the type of propositions and type ι is the type of individuals. If α and β are type symbols so is $(\alpha\beta)$; it stands for the type of functions from the type β to the type α . Brackets may be omitted with associations to the left; for example, $\gamma\beta\alpha = (\gamma\beta)\alpha$ may be considered as the type of 2-place functions whose arguments range over types α and β and whose values lie in type γ .

(2) The primitive constant symbols of the system are $N_{\sigma\sigma}$ (for negation), $A_{\sigma\sigma\sigma}$ (for disjunction), and $\Pi_{\sigma(\sigma\alpha)}$ (for generalisation). Note that ' $(x_\alpha)f_{\sigma\alpha}x_\alpha$ ' is an abbreviation for ' $\Pi_{\sigma(\sigma\alpha)}f_{\sigma\alpha}$ '.

(3) Terms with assigned types are built up from constants and variables with appropriate type suffices as follows:

(i) Application: if $A_{\alpha\beta}$ and B_β are terms of the types indicated then $(A_{\alpha\beta}B_\beta)$ is a term of type α ;

(ii) λ -abstraction: $(\lambda x_\beta A_\alpha)$ is a term of type $\alpha\beta$.

Brackets may be omitted with association to the left, and consecutive abstractions can be indicated with a single λ ; e.g., $\lambda x_\alpha y_\beta. C_\gamma$ for $(\lambda x_\alpha(\lambda y_\beta. C_\gamma))$.

I shall adopt Turing's convention that the type subscript may be dropped from constants, whether primitive or introduced as abbreviations, from bound occurrences of a variable subsequent to the binding occurrence, and from variables restricted by hypotheses. I shall also use dots for brackets, but without any particular conventions. Standard logical relations \cup , \supset , \equiv , (x_α) , $(\exists x_\alpha)$ are introduced (as abbreviations) in the usual way. Equality is defined by

$$A_\alpha = B_\alpha \quad \text{for } (f_{\sigma\alpha})(f A_\alpha \supset f B_\alpha)$$

(4) Church gives six rules of inference; these, together with his axioms 1 to 6, constitute the basic system (BC) say. It covers the rules for the typed λ -calculus and classical propositional calculus and quantification theory at all types. Church states and proves the deduction theorem and makes frequent use of it.

(5) Church introduces the natural numbers as iterators in type $u(u)$ (abbreviated to ' t'): 0_t for $\lambda f_u x_t. x$, 1_t for $\lambda f_u x_t. fx$ and so on. (Like Frege, but unlike Peano, Church counts 0 as a natural number.) The successor function is given by

$$S_{t't'} \text{ for } \lambda y_{t'} f_{ut} x_t. f(yfx).$$

The predicate 'is a natural number' is given by

$$N_{ot'} \text{ for } \lambda x_{t'} (f_{ot'})(f0 \& (y_{t'})(fy \supset f(Sy)). \supset fx).$$

The axioms 7 and 8, which together constitute an axiom of infinity, are

$$\begin{aligned} 7^t & \exists x_t, y_t. x \neq y \\ 8^t & Nx_t \& Ny_t. \supset .x_t \neq y_t \supset Sx_t \neq Sy_t \end{aligned}$$

A consequence of 7 is $(x_t) (0 \neq Sx)$. Thus these axioms and the definitions suffice to prove Peano's axioms for the natural numbers. I discuss the intuitive meaning of 8 at the end of the comment on *A Formal Theorem*.

(6) Church lists further optional axioms

$$\begin{aligned} 9^\alpha & \text{ (Axioms of Descriptions)} \\ f_{o\alpha} x_\alpha \& (y_\alpha) (f_{o\alpha} y \supset x_\alpha = y). \supset f_{o\alpha} (i_{\alpha(o\alpha)} f_{o\alpha}), \end{aligned}$$

where the description operator $i_{\alpha(o\alpha)}$ is to be added to the list of constants). Instances of these axioms are used to justify the definition of number-theoretic functions by primitive recursion.

$$\begin{aligned} 10^{\alpha\beta} & \text{ (Axioms of Extensionality)} \\ (x_\beta) (f_{\alpha\beta} x = g_{\alpha\beta} x) \supset f_{\alpha\beta} = g_{\alpha\beta} \end{aligned}$$

Church remarks that in the absence of extensionality the axioms 9^α are all independent, but that assuming extensionality they can all be derived from 9^o and 9^t . He mentions but does not name or number the proposition

$$p_o \equiv q_o \supset p_o = q_o.$$

I shall call it 10^o . It represents Frege's view that a proposition denotes its truth value. In his paper Church does not use either 10^o or $10^{\alpha\beta}$.

$$\begin{aligned} 11^\alpha & \text{ (Axioms of Choice)} \\ f_{o\alpha} x_\alpha \supset f_{o\alpha} (j_{\alpha(o\alpha)} f_{o\alpha}) \end{aligned}$$

I have followed Henkin in using different symbols for the descriptions and choice operators.

2. Turing and Church's System

In a footnote Church credits Turing with a proof that, using descriptions, $8'$ can be derived from another axiom of infinity (30'). The relation between these is discussed in Gandy [1995]. Later unpublished work of Turing is described under *Some Theorems* below. Even after he had developed his nested-type system (see *Practical Forms* below) as an alternative, he continued to admire Church's system. When I was thinking about the logical structure of physical theories he recommended it to me; by introducing a number of distinct basic types, corresponding to quantities having different physical dimensions, I could formulate dimensionality in physics very straightforwardly. And he also suggested that in order better to understand the meaning of indistinguishability (of particles) I should consider the indistinguishability of individuals in type theory. This led me to invent, by myself, the Frankel–Mostowski method for proving the independence of the Axiom of Choice (see Gandy [1952]).

3. The method of virtual types

There are a number of references to this in the surviving manuscripts but no complete account; presumably it was given in the pages missing from the manuscript of *Some Theorems*. I first give a brief account of the method.

Suppose one is particularly interested in some subset of a given type; Turing's standard example is the set of natural numbers considered as a subset defined by $N_{o\prime}$ of the type $u(u)$. Both the finding and the formalising of proofs about this set will be greatly simplified if one introduces a new basic type symbol, ρ say, and extends the definition of 'type symbol' and 'term', and the rules and axioms in the obvious way; the type ρ is called a virtual type. In order to justify the procedure one needs to provide translations of terms and proofs in the extended system into terms and proofs of the original system. The only stumbling block lies in the translation of function types in the extended system. Turing's solution (used in *Practical Forms II* for the translation from Church's system to the nested-type system) may be illustrated by considering the type $\rho\rho$. Let ρ be the subset of type α defined by $R_{o\alpha}$; then type $\rho\rho$ is translated as those functions of type $\alpha\alpha$ which for arguments in $R_{o\alpha}$ take values in $R_{o\alpha}$ and which take a standard 'nonsense' value C_α for arguments in type α but not in $R_{o\alpha}$. Using descriptions operators one easily defines the translation of terms $\lambda x_\rho A_\rho$. Of course, in translating proofs in the extended system it is necessary to prefix a formula containing free variables by a premise asserting that the variables of extended types satisfy the restrictions appropriate to those types.

In *Practical Forms II* Turing writes:

At one time the author intended to publish a paper dealing with these virtual types. The idea of virtual types however eventually led him to the practical system... Nearly all the advantages that are to be had in using virtual types with Church's system can also be obtained by using the practical system instead of Church's.

In 'the practical system' (which is the nested-type system of *Practical Forms*) virtual types are replaced by noun-classes. In *Reform* and *Practical Forms II* Turing uses, for example, the notation $(x : A)B$ for $(x)(Ax \supset Bx)$ where, for some r , $Ax \supset D^r x$ has been proved, so that A is a noun-class. This is slightly more cumbersome than the use of type subscripts in the extended system. And, despite the quoted remark, Turing remained fond of virtual types: he gave a talk about them in Oxford in 1953. Mathematicians do in fact use an informal version of virtual types: letters from certain founts or parts of founts are used for variables – whether free or bound – whose ranges are restricted by convention to specified classes; e.g., l, m, n for natural numbers, t, x, y, z for real numbers, bold face for vectors. Further letters or founts will then be introduced for variables which range over specified function-types; e.g., f, g, h for functions from real numbers to real numbers.

4. Concluding remarks

To the end of his life, Turing thought that a simple type-theory, which includes function-types, provides a particularly natural and usable foundation for ordinary mathematics, or at least for the relatively concrete parts of it. His reasons for this view are given in *Practical Forms*, *A Practical Form T* and *Reform*, and are discussed in my comments on these papers.

Even if, as is now the orthodox view, one takes set theory as the foundation for ordinary mathematics, Turing's ideas are still relevant, and can be used to simplify the complicated and sometimes tortuous set-theoretic definitions of mathematical concepts. A 'noun-class' here is a collection that is provably a set. One uses particular kinds of letter (perhaps indicated by type subscripts) as variables ranging over particular noun-classes.

A further example of his ideas is provided by those programming languages in which the types of introduced variables must always (at least in principle) be declared.

A Formal Theorem for Church's Theory of Types
(with M.H.A. Newman, J. Symbolic Logic, vol. 7 (1942),
pp. 28–33)

PREFACE

At some time in 1940, Newman became interested in exploiting the device of *typical ambiguity* to construct a system of logic. In Church's system the axioms and rules are, except for the axioms of infinity (axioms 7 and 8), stated uniformly for all types – although, of course, the type σ of propositions has a special status. Newman therefore asked whether this uniformity could be extended to axioms 7 and 8. Let us write

$$\begin{aligned} 7^\alpha &\text{ for } (\exists x_{\alpha'} y_\alpha)(x \neq y) \\ 8^\alpha &\text{ for } N_{\sigma\alpha'} x_\alpha, . \rightarrow . N_{\sigma\alpha'} y_{\alpha'} . \rightarrow . S_{\alpha'\alpha'} x_{\alpha'} = S_{\alpha'\alpha'} y_{\alpha'} \rightarrow x_{\alpha'} = y_{\alpha'} \end{aligned}$$

(In the notation of the paper, this is Y_α .)

Church had already proved that 7^α , for any type α , is a consequence of 7^ι , and that if α is one of ι', ι'', \dots then 8^α is a consequence of 8^ι . In this paper that result is extended to any type symbol α which contains an occurrence of ι .

In the event, Newman's 'forthcoming paper' [1943] deals only with expressions (terms and formulae) and not with theorems; it characterises those ('stratified') expressions in a type-free notation which are meaningful according to some specific theory of types. It makes no reference to the paper here discussed.

In an undated letter, written from his lodgings in Bletchley sometime in 1940, Turing discussed, in a rather confused way, some questions raised by Newman about Church's axioms of infinity. The letter starts:

Certainly, I should like to have a look at your 'Abs Ob' theory. You may find I am rather too vague about your questions as I am not so very well up in Church's theory as his paper. I went to a course of lectures in which the elementary side of it was developed, but not very much detail about axioms of infinity.

In the letter, Turing suggests that one might prove $8^{\sigma\iota}$ from 8^ι by using Russell's definition of finite cardinal and his axiom of infinity – an incredibly roundabout method.

The proofs in the paper are deft and economical and were made by Newman. In a later letter, dating from either 1940 or 1941, Turing writes:

I must apologise for having left all the honours of this paper to you. I am afraid I have hardly done a hand's turn. I don't think I can plead the war as an excuse as you are probably quite as busy in term as I am here.

It is worth remarking that, like Church but unlike many other contemporary logicians (e.g. Quine), Newman makes frequent use of the Deduction Theorem. Turing would also often emphasise the importance of that theorem in logical and mathematical arguments; see the discussion of *Reform* below.

Notes and corrections

- 1 Besides the use of Axiom 9 in connection with Def. 4 mentioned in the first paragraph, Axiom 9 is also required, in connection with Def. 7, to establish definition by cases.
- 2 In (b) of the third paragraph ' τ' ' is an arbitrary type symbol, not $\tau\tau(\tau\tau)$. At their second and third occurrences read ' τ' ' for ' τ ' and vice-versa.
- 3 The proof of Theorem 5 is not quite complete. As well as the inference

$$H_1, H_2, H_3 \vdash \dots$$

in the penultimate line one also needs to establish

$$H_1, a = c, H_3 \vdash \dots;$$

this is easily done.

Comment

Before 1950, say, most logicians other than Gödel and Tarski adhered to the notion of purity of method and would only use purely formal methods when dealing with formal systems. Later, of course, models were used not only to give a better understanding of formal systems, but also, via completeness theorems, to establish formal provability. Turing relished the existence of finitely constructed models for type theory without an axiom of infinity, but he did not consider what Church's axiom 8' meant in model-theoretic terms. He was, therefore, surprised (indeed, at first, incredulous) when I told him of the following:

FACT 8' holds if and only if there are *not* finitely many elements in type α .

To demonstrate this, suppose that A (type α considered as a set) has $n + 1$ (≥ 2) members, and that p is the LCM of the numbers $2, 3, \dots, n + 1$. Then it is easy to see that

$$(1) \quad f^n x = f^{n+p} x, \quad \text{for all } x \in A, \quad f : A \rightarrow A$$

and

- (2) for all $m < n$ and $Q \geq 1$, there is some $x \in A$ and $f : A \rightarrow A$ such that:
 $f^m x \neq f^{m+q} x$.

Thus the numeral $n_{\alpha'}$ (denoting the number n) does not belong to its own posterity if and only if there are more than $n + 1$ elements in type α . Together with Theorem 7 this concludes the demonstration.

It is also possible to state and prove the fact within Church's system; for details, see Gandy [1995]. This work makes use of Newman's and Turing's ideas and does not lead to a shorter proof of their theorem.

A FORMAL THEOREM IN CHURCH'S THEORY OF TYPES

M. H. A. NEWMAN and A. M. TURING

This note is concerned with the logical formalism with types recently introduced by Church [1] (and called (C) in this note). It was shewn in his paper (Theorem 26^a) that if Y^α stands for

$$N_{\alpha\alpha'}x_{\alpha'} \supset N_{\alpha\alpha'}y_{\alpha'} \supset S_{\alpha'\alpha'}x_{\alpha'} = S_{\alpha'\alpha'}y_{\alpha'} \supset x_{\alpha'} = y_{\alpha'}$$

(a form of the "axiom of infinity" for the type α), Y^α can be proved formally, from Y' and the axioms 1 to 7, for all types α of the forms ι', ι'', \dots . For other types the question was left open, but for the purposes of an intrinsic characterisation of the Church type-stratification given by one of us,¹ it is desirable to have the remaining cases cleared up. A formal proof of Y^α is now given for all types α containing ι , but the proof uses, in addition to Axioms 1 to 7 and Y' , also Axiom 9 (in connection with Def. 4), and Axiom 10 (in Theorem 9).

It was pointed out by Church that Y^α is certainly not provable for the remaining types, consisting only of \circ 's, if the system (C) augmented by the axiom

$$Q: p_\circ \equiv q_\circ \supset p_\circ = q_\circ$$

is consistent. A proof will be published by one of us (A. M. T.) that the consistency of (C)+Q follows from that of (C).

The proof of Y^α (which begins after the preparatory lemmas 1-6) contains three main steps, which may be roughly described as follows.

(a) (Theorem 7) A proof that the "axiom of infinity" (as above) is equivalent to the proposition that no number "belongs to its own posterity."

(b) (Theorem 9) A proof that if a type τ can be mapped one-one on to a part of another type τ' , a number in τ that belongs to its own posterity corresponds to a number in τ' with the same property. Hence the axiom of infinity in τ' implies the same axiom in τ .

(c) (Theorems 10 and 11) The actual construction of mappings for the pairs of types $\alpha\beta$, α and $\alpha\beta$, β (as τ' , τ).

This gives a proof of $Y^{\alpha\beta}$ if either Y^α or Y^β is proved, and hence for any type symbol α containing ι a proof of Y^α can be built up step by step from Y' .

Constant use is made of Church's "deduction theorem" (op. cit., VII):

$$\text{If } A_\circ^1, A_\circ^2, \dots, A_\circ^n \vdash B_\circ \text{ then } A_\circ^1, A_\circ^2, \dots, A_\circ^{n-1} \vdash A_\circ^n \supset B_\circ.$$

Formal proofs often begin, therefore, with a statement of "hypotheses," called $\mathcal{H}1, \mathcal{H}2, \dots$, which play the part of the A_\circ^i . In accordance with the meaning of the sign " \vdash ", all free variables in the hypotheses must remain unchanged up to the point where the deduction theorem is used.

If A_\circ^n is of the form Ca , a being a free variable, the asserted formula on the right is $Ca \supset B_\circ$, from which there follows, using Church 15^a, $\exists x Cx \supset B_\circ$. These

Received May 9, 1941.

¹ M. H. A. Newman [1].

two steps will often be taken together; used in conjunction with a proof of $\exists x Cx$ they are equivalent to the informal argument, "There is an x such that Cx : let it be a ."

The letters α , β , and γ stand for any types. Type suffixes are usually omitted from variables after their first appearance, and from constants when they are uniquely fixed by the context, since they are not needed for following the formal transformations. Each letter used as a variable carries the same type throughout the paper, and a specimen of each is placed in a list at the end. The types are sometimes restored for additional clarity or emphasis.

The following slight modifications have been made in Church's symbolism. If α is any type, α_n is defined inductively by the rules:

$$\alpha_0 \text{ is } \alpha, \quad \alpha_{n+1} \text{ is } (\alpha_n \alpha_n).$$

The logical "and" is denoted by $\&$ instead of by juxtaposition. Chains of implications, separated by commas, are used in the same way as ordinary chains of equations: $A \supset B, \supset C, \dots$, or

$$\begin{aligned} A &\supset B, \\ &\supset C, \\ &\dots \dots \end{aligned}$$

Commas are similarly used in chains of equations. Brackets and bold dots are used as in Church [1], save that the distinctive use of square brackets is not maintained; and brackets at the end of a formula, or immediately before a comma, are usually omitted: all outstanding right-brackets are to be supplied at these points.

$$\text{DEF. 1. } s_{\alpha_3 \alpha_2} \rightarrow \lambda m_{\alpha_2} \lambda n_{\alpha_2} \lambda f_{\alpha_1} \lambda z_{\alpha} (mf(nfz).$$

$$1. \quad s(Sm)n = S(smn).$$

$$\begin{aligned} s(Sm)n &= \lambda f \lambda z (f(mf(nfz, \\ &= \lambda f \lambda z (f(smn) fz, \\ &= S(smn). \end{aligned}$$

$$2. \quad Nn \supset \lambda f \lambda z (nfz) = n.$$

$$\begin{aligned} \lambda f \lambda z (0 fz) &= \lambda f \lambda z z, = 0. \\ \lambda f \lambda z (Snfz) &= \lambda f \lambda z (f(nfz)), = Sn. \end{aligned}$$

$$3. \quad Nm \& Nn \supset N(smn).$$

$$s0n = \lambda f \lambda z (0f(nfz)), = \lambda f \lambda z (nfz), = n.$$

By 1, $Nm \& Nn \supset N(smn) \supset N(s(Sm)n)$, and the result follows by induction.

$$4. \quad Nm \& Nn \supset smn = snm.$$

$$\begin{aligned} (a) \quad sm0 &= \lambda f \lambda z (mf(0fz, \\ &= \lambda f \lambda z (mfz), = s0m \end{aligned} \quad (\text{proved in 3}).$$

$$(b) \quad s1m = sm1 \vdash$$

$$\begin{aligned}
 s1(Sm) &= \lambda f \lambda z (f(Smfz, \\
 &= \lambda f \lambda z (f(f(mfz, \\
 &= \lambda f \lambda z (f(s1mfz, \\
 &= \lambda f \lambda z (f(sm1fz, \quad (\text{using the premiss}) \\
 &= \lambda f \lambda z (f(mf(fz, \\
 &= \lambda f \lambda z (Smf(fz, \\
 &= s(Sm)1.
 \end{aligned}$$

Hence by induction, $Nm \supset s1m = sm1$.

$$(c) \quad Nn, smn = snm \vdash$$

$$\begin{aligned}
 s(Sm)n &= \lambda f \lambda z (Smf(nfz, \\
 &= \lambda f \lambda z (f(mf(nfz, \quad (\text{Def. of } S) \\
 &= \lambda f \lambda z (f(smnfz, \\
 &= \lambda f \lambda z (f(snmfz, \quad (\text{using the second premiss}) \\
 &= \lambda f \lambda z (f(nf(mfz, \\
 &= \lambda f \lambda z (s1nf(mfz, \\
 &= \lambda f \lambda z (sn1f(mfz, \quad (\text{by (b)}) \\
 &= \lambda f \lambda z (nf(f(mfz, \\
 &= \lambda f \lambda z (nf(Smfz, \\
 &= sn(Sm).
 \end{aligned}$$

Hence the theorem follows by induction.

$$5. \quad Nm_{\alpha_2} \& Nn_{\alpha_2} \& m \neq n \supset \exists r_{\alpha_2} . Nr \& . m = sn(Sr) \vee n = sm(Sr).$$

(a) Since it is readily proved by induction that

$$Nn \& 0 \neq n \supset \exists r . Nr \& n = Sr,$$

and since $n = s0n$ was proved in 3, we have

$$Nn \& 0 \neq n \supset \exists r . Nr \& n = s0(Sr).$$

$$\begin{aligned}
 (b) \quad \mathcal{H}1. \quad &Na_{\alpha_2} \& Nc_{\alpha_2} \& Ne_{\alpha_2}. \\
 \mathcal{H}2. \quad &a \neq c. \\
 \mathcal{H}3. \quad &a = sc(Se) \vee c = sa(Se).
 \end{aligned}$$

We have:

$$\begin{aligned}
 a = sc(Se) &\supset Sa = sc(S(Se) \quad (\text{by 1 and 4}). \\
 c = sa(Se) &\supset . c = sa(Se) \& . e = 0 \vee \exists s . Ns \& e = Ss, \\
 &\supset . c = Sa \vee \exists s . Ns \& c = sa(S(Ss, \\
 &\supset . c = Sa \vee \exists s . Ns \& c = s(Sa)(Ss).
 \end{aligned}$$

Hence $\mathcal{H}1, \mathcal{H}2, \mathcal{H}3 \vdash Sa = sc(S(Se)) \vee Sa = c \vee \exists s . Ns \& c = s(Sa)(Ss)$. On eliminating the hypotheses the theorem follows by induction.

$$6. \quad (x_{\beta}) \sim g_{\alpha_2 \beta} x 0 \& \exists n \exists x (Nn \& gx n) \supset \exists r_{\alpha_2} . Nr \& \sim \exists x (gxr) \& \exists x (gx(Sr).$$

This is merely a form of the Principle of Induction, and the formal proof may be omitted.

DEF. 2. $X_{\alpha_2\alpha_2}^{\alpha} \rightarrow \lambda m_{\alpha_2} \lambda n_{\alpha_2} . Nm \ \& \ Nn \ \& . \ S(smn) = n.$

DEF. 3. $Y_{\alpha_2\alpha_2}^{\alpha} \rightarrow \lambda m_{\alpha_2} \lambda n_{\alpha_2} . Nm \ \& \ Nn \supset . \ Sm = Sn \supset m = n.$

Thus Theorem 26^a of Church [1] is equivalent under a law of the propositional calculus and familiar laws of quantifiers to $(m)(n)Y^{\alpha}mn$, and the two theorems may conveniently be thought of as the same.

7. $(m)(n) \sim X^{\alpha}mn \equiv (m)(n)Y^{\alpha}mn.$

First part. $\mathcal{H}1. \ Na \ \& \ Nc.$

$\mathcal{H}2. \ (n) \sim X^{\alpha}na.$

$\mathcal{H}3. \ Sa = S(sc(Sa).$

$$\begin{aligned} \text{From } \mathcal{H}3, \ sc(Sa) &= sc(S(sc(Sa, \\ &= S(sc(sc(Sa. \end{aligned}$$

Thus,

$$X^{\alpha}c(sc(Sa,$$

and hence,

$$\exists n(X^{\alpha}n(sc(Sa.$$

Combined with $\mathcal{H}2$ this gives $a \neq sc(Sa)$, and hence $\mathcal{H}1, \mathcal{H}2, \mathcal{H}3 \vdash \exists m \exists n . Nm \ \& \ Nn \ \& \ m \neq n \ \& \ Sm = Sn$. Hence upon eliminating the hypotheses,

$$\exists r(Nr \ \& \ \sim \exists n(X^{\alpha}nr) \ \& \ \exists n(X^{\alpha}n(sr))) \supset \exists m \exists n \sim Y^{\alpha}mn.$$

Now $(n) \sim X^{\alpha}n0$, by 3 and Church 25^a. Therefore, from 6 (observing that $X^{\alpha}mn \supset . Nn \ \& \ X^{\alpha}mn$),

$$\exists m \exists n(X^{\alpha}mn) \supset \exists m \exists n \sim Y^{\alpha}mn.$$

Second part. $\mathcal{H}1. \ Na \ \& \ Nc \ \& \ Ne.$

$\mathcal{H}2. \ a \neq c.$

$\mathcal{H}3. \ Sa = Sc.$

$\mathcal{H}4. \ c = sa(se) \ \& \ a = sc(se).$

Since from $\mathcal{H}1, \mathcal{H}3, \mathcal{H}4$ there follows (by 1 and 4)

$$Sa = S(se(Sa)) \ \& \ Sc = S(se(Sc)),$$

we have $\mathcal{H}1-\mathcal{H}4 \vdash \exists m \exists n(X^{\alpha}mn)$, and hence eliminating the hypotheses and using 5 we obtain

$$\exists m \exists n \sim Y^{\alpha}mn \supset \exists m \exists n(X^{\alpha}mn)$$

DEF. 4. $\omega_{\beta_1\alpha_1(\beta\alpha)} \rightarrow \lambda h_{\beta\alpha} \lambda f_{\alpha_1} \lambda x_{\beta} (\eta y_{\beta}) .$

$$(u_{\alpha})(x = hu \supset y = h(fu)) \ \& . \ (v_{\alpha})(x \neq hv) \supset y = x.$$

DEF. 5. $\Omega_{\alpha(\beta\alpha)}^{\beta\alpha} \rightarrow \lambda h_{\beta\alpha} (u_{\alpha})(v_{\alpha}) . hu = hv \supset u = v.$

By Church's Axiom 9, $\Omega^{\alpha\beta}h \supset . \omega hf(hu) = h(fu).$

DEF. 6. $K_{\alpha\beta_2\alpha_2(\beta\alpha)}^{\beta\alpha} \rightarrow \lambda h_{\beta\alpha} \lambda m_{\alpha_2} \lambda p_{\beta_2} (f)(u) . p(\omega hf)(hu) = h(mu).$

$$8. (Np_{\beta_2} \& \Omega^{\beta\alpha}h) \supset \exists m_{\alpha_2} \cdot Nm \& K^{\beta\alpha}hmp.$$

$$\mathcal{H}1. Na_{\alpha_2} \& Nb_{\beta_2}.$$

$$\mathcal{H}2. K^{\beta\alpha}k_{\beta\alpha}a_{\alpha_2}b_{\beta_2}.$$

$$\mathcal{H}3. \Omega^{\beta\alpha}k_{\beta\alpha}.$$

$$Sb(\omega kf)(ku) = \omega kf(b(\omega kf)(ku),$$

$$= \omega kf(k(afu), \quad (\mathcal{H}2)$$

$$= k(f(afu), \quad (\mathcal{H}3)$$

$$= k(Safu),$$

i.e., $K^{\beta\alpha}k(Sa)(Sb)$. Thus, eliminating the hypotheses,

$$Na \& Nb \& \Omega^{\beta\alpha}k \supset . K^{\beta\alpha}kab \supset K^{\beta\alpha}k(Sa)(Sb).$$

Hence,

$$Nb \& \Omega^{\beta\alpha}k \supset (\exists m \cdot Nm \& K^{\beta\alpha}kmb) \supset \exists m \cdot Nm \& K^{\beta\alpha}km(Sb).$$

Now $(f)(z) \cdot 0_{\beta_2}(\omega kf)(kz) = k(0_{\alpha_2}fz)$, i.e., $K^{\beta\alpha}k0_{\alpha_2}0_{\beta_2}$. Therefore 8 follows by induction on p .

$$9. \exists h_{\beta\alpha}\Omega^{\beta\alpha}h_{\beta\alpha} \supset . (m_{\alpha_2})(n_{\alpha_2})Y^{\alpha}mn \supset (p_{\beta_2})(q_{\beta_2})Y^{\beta}pq.$$

$$\mathcal{H}1. Na_{\alpha_2} \& Nb_{\beta_2} \& Nc_{\alpha_2} \& Nd_{\beta_2}.$$

$$\mathcal{H}2. \Omega^{\beta\alpha}k_{\beta\alpha}.$$

$$\mathcal{H}3. K^{\beta\alpha}kab.$$

$$\mathcal{H}4. K^{\beta\alpha}kcd.$$

$$\mathcal{H}5. d = S(sbd).$$

$$k(cfz) = d(\omega kf)(kz), \quad (\mathcal{H}4)$$

$$= S(sbd)(\omega kf)(kz), \quad (\mathcal{H}5)$$

$$= \omega kf((sbd)(\omega kf))(kz), \quad (\text{by 4})$$

$$= \omega kf(d(\omega kf))(b(\omega kf))(kz), \quad (\mathcal{H}3)$$

$$= \omega kf(d(\omega kf))(k(afz), \quad (\mathcal{H}3)$$

$$= \omega kf(k(cf(afz), \quad (\mathcal{H}4)$$

$$= k(f(cf(afz, \quad (\mathcal{H}2)$$

$$= k(s(Sc)afz, \quad (\text{def. of } s)$$

$$= k(S(sca)fz) \quad (\text{by 1}).$$

Hence $(f)(z) \cdot cfz = S(sca)fz$ follows on using $\mathcal{H}2$, and therefore, by Church's Axiom 10, $c = S(sca)$; and from this, $\exists m \exists n (X^{\alpha}mn)$. Eliminating the hypotheses and using 8, we obtain

$$\exists h_{\beta\alpha}(\Omega^{\beta\alpha}h_{\beta\alpha}) \supset . \exists p_{\beta_2}\exists q_{\beta_2}(X^{\beta}pq) \supset \exists m_{\alpha_2}\exists n_{\alpha_2}(X^{\alpha}mn).$$

$$10. \exists \bar{h}_{(\alpha\beta)\alpha} \cdot \Omega^{(\alpha\beta)\alpha}\bar{h}_{(\alpha\beta)\alpha}.$$

Since $(\lambda z_{\alpha}\lambda x_{\beta}z_{\alpha})u_{\alpha} = \lambda x_{\beta}u_{\alpha}$, we have

$$(\lambda z_{\alpha}\lambda x_{\beta}z_{\alpha})u_{\alpha} = (\lambda z_{\alpha}\lambda x_{\beta}z_{\alpha})v_{\alpha} \supset u_{\alpha} = v_{\alpha},$$

$$\text{i.e., } \Omega^{(\alpha\beta)\alpha}(\lambda z_{\alpha}\lambda x_{\beta}z_{\alpha}).$$

DEF. 7. $Z_{\alpha\beta\alpha\alpha} \rightarrow \lambda u_\alpha \lambda v_\alpha \lambda x_\beta \lambda y_\beta (\iota z_\alpha) \ . (x=y \supset z=u) \ \& \ (x \neq y \supset z=v).$

11. $\exists \bar{k}_{(\alpha\beta)\beta} \ . \Omega^{(\alpha\beta)\beta} \bar{k}_{(\alpha\beta)\beta}.$

$\mathcal{K}1. \ s_\alpha \neq t_\alpha$

$\mathcal{K}2. \ Z_{\alpha\beta\alpha\alpha} s_\alpha t_\alpha \bar{c}_\beta = Z_{\alpha\beta\alpha\alpha} s_\alpha t_\alpha \bar{d}_\beta.$

$(\bar{c}=w_\beta \supset Zst\bar{c}w=s) \ \& \ (\bar{d} \neq w \supset Zst\bar{d}w=t).$

Hence,

$$\begin{aligned} \bar{c} \neq \bar{d} &\supset \exists w \ . \bar{c}=w \ \& \ \bar{d} \neq w, \\ &\supset \exists w \ . Zst\bar{c}w=s \ \& \ Zst\bar{d}w=t, \\ &\supset s=t \end{aligned} \quad (\mathcal{K}2).$$

Eliminating the hypotheses, we obtain

$$\exists s \exists t \ . \ s \neq t \supset \exists s \exists t (\bar{c} \neq \bar{d}) \ . \ (Zst)\bar{c} = (Zst)\bar{d} \supset \bar{c} = \bar{d}.$$

Hence, using 27^a of Church [1], 11 follows.

THEOREM. *If α is a type symbol containing ι , the formal theorem $(m_{\alpha_2})(n_{\alpha_2})Y^\alpha mn$ is provable.*

For if α is ι , the formal theorem is Axiom 8. If α is $(\beta\gamma)$, either β or γ contains ι , and by an inductive hypothesis on the length of α the truth of the theorem for β or γ may be assumed. Then the truth of the theorem for α follows by 10 or 11, combined with 9.

List of letters carrying fixed type suffixes: $a_{\alpha_2}, b_{\beta_2}, c_{\alpha_2}, \bar{c}_\beta, d_{\beta_2}, \bar{d}_\beta, e_{\alpha_2}, f_{\alpha_1}, g_{\alpha_2\beta}, h_{\beta\alpha}, \bar{h}_{(\alpha\beta)\alpha}, k_{\beta\alpha}, \bar{k}_{(\alpha\beta)\beta}, m_{\alpha_2}, n_{\alpha_2}, p_{\beta_2}, q_{\beta_2}, r_{\alpha_2}, s_\alpha, t_\alpha, u_\alpha, v_\alpha, w_\beta, x_\beta, y_\beta, z_\alpha, Z_{\alpha\beta\alpha\alpha}, \omega_{\beta_1\alpha_1(\beta\alpha)}.$

For any type symbols α and β , X^α , Y^α , $\Omega^{\beta\alpha}$, and $K^{\beta\alpha}$ are abbreviations for $X_{\alpha_2\alpha_2}^\alpha$, $Y_{\alpha_2\alpha_2}^\alpha$, $\Omega_{\alpha_2(\beta\alpha)}^{\beta\alpha}$, and $K_{\alpha_2\alpha_2(\beta\alpha)}^{\beta\alpha}$, respectively.

REFERENCES

- A. Church [1]. *A formulation of the simple theory of types*, this JOURNAL, vol. 5 (1940), pp. 56-68.
 M. H. A. Newman [1]. *Stratified systems of logic*. Forthcoming.

ST. JOHN'S COLLEGE, CAMBRIDGE
 KING'S COLLEGE, CAMBRIDGE

This Page Intentionally Left Blank

The use of dots as brackets in Church's system

(J. Symbolic Logic, vol. 7 (1942), pp. 146–156)

PREFACE

Peano (in his *Calcolo geometrico* 1888) was, I believe, the first writer to use groups of dots instead of brackets in order to make the structure of complex formulae more easily graspable; a more elaborate version of his system was used in *Principia Mathematicae*. Curry [1937] pointed out that this system is not well suited to formulae which are built up in an iterative way; he elaborated on an idea of Church's in order to give an alternative set of conventions for the use of dots. Turing was, presumably, already familiar with Curry's paper when he started building up a collection of conventions in his work on Church's system. Mostly, these conventions differ from the ones used in *A Formal Theorem* (preceding article).

In Letter V, Turing wrote:

I have been using my dot notation, and have found it quite satisfactory, except as regards dots adjacent to λx_α ; I have had to introduce the rule that dots adjacent to (λx_α) are on a par with ones adjacent to (x_α) , whilst ones adjacent to λx_α remain as before. The difficulty about putting in type suffixes has been eliminated by omitting them on bound variables (except at the binding place), e.g., $(x_\alpha) f_{\sigma\alpha} x$, and also on 'restricted' variables, i.e. on variables which may not be substituted for on account of the conditions of the Deduction Theorem. This seems to be healthy: it reminds one that the variables are restricted, and also encourages one to look at the hypotheses.

(For the use of (λx_α) see **Note 1** immediately below.)

Note 1

As explained later, the difference between (λx_α) and λx_α is that Turing writes $(\lambda x_\alpha) B_\beta$ if β begins with o , $\lambda x_\alpha B_\beta$ otherwise. Since $(x_\alpha) B_o$ is actually an abbreviation for $\Pi_{o(o\alpha)}(\lambda x_\alpha B_o)$ this convention is required for consistency.

Note 2

p. 147, line 14. It is not quite clear what Turing means here by an 'ordering of the points'. In the application to Church's system, Turing assigns a power to each point in such a way that the following conditions are satisfied:

- (i) The powers are linearly ordered,
- (ii) Two similarly facing points may have the same power (see, e.g., the points after [m] and [p] in example (iii) on p. 155 of Turing's paper).

- (iii) If the other conventions assign equal powers to two oppositely facing points, then the left facing one is given a higher power than the right facing one.

These conditions impose a pre-linear ordering, \leqslant , say, on the points such that

- (A) if α and β are oppositely facing then either $\alpha < \beta$ or $\beta < \alpha$, where $\alpha < \beta$ is short for: $\alpha \leqslant \beta$ & not $\beta \leqslant \alpha$.

Sometimes, however, in the arguments that follow, Turing seems to assume that the ordering \leqslant is actually linear. Certainly, if condition (A) is not satisfied then the first form of rule (p. 147, line 23) cannot be applied without ambiguity, and the scopes of brackets introduced by the second form of rule (p. 147, line b8) may overlap. For example, a projected formula $M.x.Dy$ could stand for $(Mx)Dy$ or $M(xDy)$ and would by the second form of rule be bracketed as $(M[x]Dy)$.

Curry avoided this difficulty by adopting the convention that if all the other conventions assigned equal powers to two points then the rightmost one should have the higher power. This linearly orders the points. It is not hard to see that this leads to exactly the same bracketing as do the conditions (i)–(iii) above.

Note 3

p. 150, line b9. The argument that follows is typical of Turing's way of doing things with his bare hands; the reader too will need rolled up sleeves to follow it! I give a slightly more perspicuous version. As Turing shows, if (the bracket at) β is moved to the left then β cannot coincide with δ ; hence, between δ and β there is a right-facing point ε of power not less than that of δ . If the brackets at γ , δ arise from γ (so that the point at γ is removed), then all the right-facing points from γ up to δ have powers less than δ , and hence less than ε ; so β cannot be moved in this case. If the brackets at γ , δ arise from δ (so that the point at γ lies outside the inserted brackets) then the powers of the right-facing points between these brackets are less than the power of γ and β is not moved in this case either.

Commentary and Discussion

1) As observed in **Note 2**, Turing's conventions, in general, only differ from those introduced by Curry in unimportant ways, and his first and second forms of rule are almost the same as Curry's Rule I and Rule II. In a reversal of their typical roles, it is Curry who merely remarks that the equivalence is 'readily shown' and Turing who provides a detailed proof. It is reasonable to suppose that his work at Bletchley Park was such that he did not feel like working on hard problems, but enjoyed the comparative minutiae of his work with Church's system as a form of relaxation.

2) Turing's attitude to conventions was ambivalent. On the one hand, he often argued in favour of strict conventional rules about unimportant matters – for

example *dress* or *notation* – because the rules relieved one of the strain of worrying about trifles. Indeed, he could be mildly obsessive in his adherence to certain forms and rituals. (He once stuttered with indignation when I omitted the final downstroke in writing ‘ μ ’.) On the other hand, his temperament was bohemian rather than conformist. I remember him being very aggrieved by some poker players who regarded it as very bad form to leave the game immediately after winning a hand; and, as his mother’s autobiography makes plain, he was often in trouble at school because of his impatience with rules and conventions which prevented him from getting straight on with the job in hand. His short cuts and direct attacks on problems or situations, sometimes dazzling, sometimes merely perverse, disconcerted more orthodox people.

This ambivalence is reflected in his views on notation and in the present paper. An abbreviated formula is at first subjected (on p. 146–7) to the rather stringent conditions (a), (b) and (c), but these are later relaxed; and it should be noted that he explicitly renounces the use of a *unique* abbreviated form for a given formula (see p. 155 for examples). As Turing explains in discussing examples at the end of this paper, his objective in using dots was to make it as easy as possible to perceive the structure of a long formula at a glance. (He objected strongly to Lukasiewicz’s bracket-free notation because it fails in this respect.) He also disliked conventions which have to be learnt by heart: the only artificial convention in this paper is that left-facing is stronger than right-facing. Occasionally, he would enjoy reciting some mathematical formula which had to be learnt by heart. More often, however, he could not be bothered with memorising things, knowing that if required he could work them out. This cavalier attitude could be a serious handicap. I remember once he had devised a particularly simple programme for carrying out λ -conversions on a computer. I objected that it did not take account of the restriction in converting $(\lambda x M)N$ that “the bound variables of M be distinct from x and the free variables of N ”. “Good Lord,” he replied, “how right you are; knowing that by heart is worth £100 a year!”

It is therefore not surprising that in his work on the theory of types he did not rely too heavily on the conventions of this paper. One can normally read the formulae without even being aware of the details of the conventions.

THE USE OF DOTS AS BRACKETS IN CHURCH'S SYSTEM

A. M. TURING

Any logical system, if its use is to be carried beyond a rather elementary stage, needs powerful conventions about abbreviations: in particular one usually wants to modify the bracketing so as to make the formulae more readable, and also possibly shorter. The present note has been written in the belief that Church's formulation of the simple theory of types¹ is particularly suitable as a basis for work on that theory, and that it is therefore worth while introducing special conventions which take into account the needs of this particular system. The conventions which I shall describe are ones which I have used a good deal myself, and have always found adequate. I intend to make use of them in forthcoming papers.² They may be regarded as an extension of Curry's conventions.³

I shall begin with a general discussion of punctuation by means of groups of dots. This general theory is applicable, with some modifications, to Russell's,⁴ Quine's,⁵ and Curry's³ bracketing systems as well as to the present one.

General bracketing theory. We consider a logical system in which every formula is either:

An irreducible formula (or *token* in Curry's terminology).

Of form $R(A)$ where R is a monadic operator and A a formula.

Of form $(A)S(B)$ where S is a dyadic operator and A and B are formulae.

We need not of course enquire further into the nature of the irreducible formulae, monadic operators, and dyadic operators, but to fix our ideas we may think of irreducible formulae as consisting of a single letter with suffixes etc., e.g. x_α , $J_{\beta(\alpha)}^{\alpha}$. Typical of monadic operators would be \sim , $[\exists x_\alpha]$ and of dyadic operators \supset and $=$. The formulae in this sense will be described in future as *unabbreviated formulae*: the word 'formula' without qualification will be liable to be used of various kinds of series of symbols.

We may also recognise another kind of formulae which we call *abbreviated formulae* and which consist of series of symbols which are irreducible formulae, brackets, monadic and dyadic operators, and a new kind of symbol called a point, which may be thought of as a group of dots. To be an abbreviated formula the series of symbols must satisfy the conditions:

(a) The brackets must be properly paired, i.e., if we go on removing pairs of brackets which face each other and have no other brackets between them there should eventually be no brackets left. The brackets appearing in an abbreviated formula will often be described as 'explicitly shown brackets.'

Received June 17, 1942.

¹ Alonzo Church, *A formulation of the simple theory of types*, this JOURNAL, vol. 5 (1940), pp. 56-68.

² A. M. Turing, *Some theorems about Church's system, and The theory of virtual types*, forthcoming.

³ H. B. Curry, *On the use of dots as brackets in logical expressions*, this JOURNAL, vol. 2 (1937), pp. 26-28.

⁴ Whitehead and Russell, *Principia mathematica*, vol. 1, pp. 9-11.

⁵ W. V. Quine, *Mathematical logic* (New York 1940), pp. 37-42.

(b) Of a pair of brackets one must occur adjacent to an operator and one not. The expression 'adjacent to an operator' is used here and elsewhere to mean 'adjacent to a dyadic operator or adjacent to and on the right of a monadic operator.'

(c) If in the formula we replace dyadic operators by 'D', monadic operators by 'M', irreducible formulae by 'x' and points by '·', calling the result the 'projected formula,' then the first symbol of a projected abbreviated formula must be '(', 'x', or 'M' and the last, ')' or 'x'. A pair of consecutive symbols in the projected formula must be 'x)', '(x, ')', 'M()', 'D(' or ')D', '(M' or '()' or else part of one of the following series of three: 'x:D', 'D:x', 'M:x', '·:D', 'D:·', 'M:·', 'D:M', 'M:M': in the latter case the whole series of three symbols must be part of the projected formula.

We want one and only one formula to correspond to each abbreviated formula. Such a correspondence is defined below in terms of an ordering of the points. I shall follow Russell's terminology and speak of the earlier of two points in the ordering as being of *higher power* than the other. Curry uses the expression 'senior to' and Quine, whose points are called 'joints,' uses 'looser than.' The power of a point may depend on any formal relationships between the point and the formula it occurs in, and varies from system to system.

The rule for replacing the abbreviated formula by the unabbreviated may be put into two forms, of which the first is the more natural theoretically, and the second, which seems rather arbitrary, is the easier to apply.

First form of rule. The rule operates by reducing the number of points in the formula whose unabbreviated form is to be found.

Suppose first that the formula has explicit brackets, e.g. that it is of form $A(B)C$, where A , B , C are not required to be formulae in any special sense, but just rows of symbols, and the brackets shown are properly paired. Then the unabbreviated form of $A(B)C$ may be obtained from the unabbreviated forms E of AwC and F of B by substituting (F) for w in E . The symbol w is to be some symbol not occurring in A or C . In other words the interior of an explicitly shown bracket is to be worked out as if it were a whole formula, and the part of the formula outside the bracket is to be worked out as if the bracketed part were a single letter.

If the formula has no explicitly shown brackets we find the point of highest power and replace it by a bracket. This bracket is to be right facing if the point is right facing, i.e., if it is on the right of its operator: similarly the bracket is left facing if the point is left facing. Another bracket, oppositely facing, must be put at one end of the formula to balance the first.

Second form of rule. We first define the *enclosing brackets* of a symbol other than an explicitly shown bracket. They are paired explicitly shown brackets, enclosing the symbol in question, but not enclosing any other pair of brackets which enclose the symbol. If the enclosing brackets are always to be defined there must be a pair of brackets enclosing the whole formula. We imagine these supplied.

To find the unabbreviated formula we clearly have to replace each point by a similarly facing bracket, and to put in a balancing bracket somewhere. The

interval from the point to the balancing bracket is called the *scope* of the point: in reckoning scopes, points and brackets will be neglected, so that for instance if two similarly facing points are to have their partnering brackets immediately following one another their scopes will be regarded as ending at the same place. The rule for determining the scope is that it is to be as short as possible, subject to the following *scope condition*:

The balancing bracket β of a point π is either adjacent to one of the enclosing brackets of π , or else to some point ρ facing oppositely to π and having the same enclosing brackets as π , in which case β must be on the side of ρ which is nearer to π . The point ρ must be of higher power than π or any point between ρ and π facing similarly to π and having the same enclosing brackets as π .

Equivalence theorem. There are three things to be proved about these rules:

(i) When we use the first rule it does not matter in what order the pairs of explicit brackets are taken.

(ii) The result of applying the first rule to an 'abbreviated formula' (satisfying by definition conditions (a), (b), (c) above) is to give us an 'unabbreviated formula' as originally defined.

(iii) The two rules are equivalent.

To prove (i) let $A(B)C$ be one of the shortest formulae for which the result of applying the rule is not unique. We are justified in assuming that explicit brackets occur for otherwise the first step in applying the rule is uniquely determined and consists in introducing brackets. Whatever transformation we apply to the formula it remains of the form $A'(B')C'$ where $A'wC'$ is obtained from AwC and B' from B by a (possibly incomplete) application of the rule. In particular this is true of the final result of applying the rule. In this case $A'wC'$ contains no points: it is therefore the final result of applying the rule to AwC and since AwC is shorter than $A(B)C$ the formula must be unique. Similarly B' is unique, and therefore $A'(B')C'$ is unique.—The word 'shortest' as used in this argument must be interpreted as 'having the smallest number of symbols, points however being reckoned as two symbols.'

To prove (ii) it is sufficient to show that the application of the transformations described in the rule always leaves us with an abbreviated formula, and that if an abbreviated formula has no points then it is an unabbreviated formula. The transformations always consist of the removal of a point and the introduction of a pair of brackets. The brackets have no other brackets between them, so that the brackets remain properly paired, i.e., (a) remains satisfied. One of the brackets replaces a point, and therefore, by (c) applied to the original formula, is adjacent to an operator. The other bracket is put in either at the end of the formula or adjacent to a similarly facing bracket, facing away from it. It cannot be adjacent to an operator, for if it were there would have been an operator adjacent to the end of the formula, or to a bracket facing towards it, in the original formula, contradicting (c). This shows that (b) remains true. To show that (c) remains true we have only to notice that when we replace points by similarly facing brackets in the admissible combinations the results are made up of admissible combinations, and that admissible combinations always

result when a bracket is introduced at the end of the formula or adjacent to a similarly facing bracket.

To prove the second requirement let us see what condition (c) amounts to when there are no points in the formula. The allowable pairs of symbols in the projected formula are 'x)', 'M(', 'D(', ')D', ')'), '(x', '(M', '()' and a formula must start with '(', 'M', or 'x' and end with 'x' or ')'. If it starts with 'x' it can only continue with ')' and this bracket can have no partner: i.e., if the projected formula starts with 'x' then 'x' is the whole of it. If it starts with 'M' it continues with '(', and this bracket has a partner, so that the whole is of form $M(\mathbf{A})\mathbf{B}$, and by (b) of form $M(\mathbf{A})$. If the formula starts with '(' this has a partner which by (b) is adjacent to an operator: i.e., the formula is of form $(\mathbf{A})DB$ and therefore of form $(\mathbf{A})D(\mathbf{C})E$. Applying (b) we see it is of form $(\mathbf{A})D(\mathbf{C})$. Thus we have shown that abbreviated formulae without points are always either irreducible formulae or of one of the forms $R(\mathbf{A})$ or $(\mathbf{A})S(\mathbf{B})$, where R is a monadic and S a dyadic operator. The formulae \mathbf{A} and \mathbf{B} necessarily satisfy the conditions (a), (b), (c) since the whole formula satisfies them, and the symbols allowed at the ends of a formula by (c) are just the ones which may follow a right facing bracket or precede a left facing bracket: these formulae are therefore themselves 'abbreviated formulae.' An induction over the length of the formula will now prove that every abbreviated formula without points is an unabbreviated formula, as required.

To prove (iii) notice that the second rule agrees with the first as regards the replacement of the points of highest power, for with either rule we may suppose that the enclosing brackets of the point to be replaced are at the ends of the formula. It will therefore be sufficient to prove that the order of replacement of two points may be interchanged when we are using the second rule.

The case when the two points did not originally have the same enclosing brackets is trivial, for then the replacement of the one point does not alter the set of symbols having the same enclosing brackets as the other, and therefore does not alter its scope. We may therefore suppose that the enclosing brackets of both points are at the ends of the formula. We may also suppose that there are no other brackets in the formula, for if any pair of brackets, together with what is between them, is replaced by a single letter, the scope of neither of the points is altered.

The scopes of two points can never be strictly overlapping. Suppose that the scope of one point is limited by brackets α and β of which α is the one further to the left, and the other by γ and δ of which γ is to the left; also that α is to the left of β and that the scopes strictly overlap, so that the brackets form a figure like this:

$$(\dots(\dots)\dots)$$

$\alpha \quad \gamma \quad \beta \quad \delta$

The points from which these brackets arise can be either at α and γ , or at α and δ , or at β and γ , or at β and δ . The consideration of the last alternative can be omitted as it is the same as the first apart from interchange of left and right.

In the case that the points are at α and γ the brackets α, β must satisfy the scope condition, so that the point at β must be of higher power than those at α and γ or any right facing point between α and β ; in particular it is of higher power than those at γ and between γ and β , and therefore by the scope condition the bracket δ partnering γ must have the same position as β , in which case the scopes do not strictly overlap. Next suppose that the points are at α and δ . Then applying the scope condition to the brackets α and β we find that the point at β is stronger than that at γ , and this means that the scope condition cannot be satisfied for a point at δ whose partnering bracket is at γ . Finally suppose that the points are at β and γ . Applying the scope condition to γ and δ we see that either γ or some right facing point between it and β is of higher power than β : but if this is so the scope condition cannot be satisfied for α and β .

This completes the proof that the scopes of two points can never be strictly overlapping, and we now apply it to the interchange of order of removal of brackets under the second rule. Suppose that the scope of the first point is from α to β , the point itself being at α , which we suppose to be to the left of β , and the scope of the second from γ to δ ; γ being to the left of δ , but no assumption being made as to whether the point was at γ or δ . We wish to show that the scope of the first point as calculated by the scope condition is unaltered if the other point is replaced by its brackets γ, δ . To fix our ideas we suppose that the scopes α to β and γ to δ are as calculated before either pair of brackets has been put in. The scope of the first point is certainly unaltered by the replacement of the second in the case that the scopes do not overlap at all, for then neither the points within the interval α to β , nor the left facing point (or possibly bracket) at β can be altered by the introduction of γ and δ , and the application of the scope condition gives exactly the same result for the position of β . As the scopes cannot strictly overlap we must suppose that either the interval α to β is wholly contained in the interval γ to δ or wholly contains it. In the first case the data for the application of the scope condition to the bracket β are again not relevantly altered. If the interval γ to δ is wholly contained in the interval α to β we consider separately the possibilities that β might be moved farther to the right or farther to the left by the introduction of γ and δ . To show that β is not moved farther to the right it will be sufficient to show that the interval still satisfies the scope condition. This is certainly the case, for the effect of the introduction of γ and δ , so far from introducing new right facing points is to enclose some in brackets, thereby as it were disqualifying them, and also to remove the point from which γ and δ themselves arose. To show that β is not moved farther to the left we have to show that there can be no left facing points ρ between α and β which satisfy the scope condition. Such a point would certainly have to be between δ and β , for if it were between γ and δ it would not have the same enclosing brackets as α , and if it were between α and γ the position of β would have been at ρ regardless of whether the brackets γ and δ had been put in or not. If ρ between δ and β satisfies the scope condition, then in the original formula there must have been a right facing point σ either at γ , or in the interval γ to δ , which was more powerful than ρ and less powerful than the point at β . However, as the scope of the bracket γ, δ , if

it arises from a point at γ , extends only as far as δ , there must have been a point at δ more powerful than σ and therefore than ρ and all right facing points in the interval α to γ . The original position of β would therefore have been the position of δ . If on the other hand the brackets γ and δ arise from a point at δ , then ρ must have been less powerful than some right facing point σ in the interval γ to δ without the alternative of σ being γ itself. We may suppose that σ is the right facing point of highest power in the interval γ to δ . But then as the bracket from δ extends as far as γ , either the point at δ or some left facing point τ in the interval σ to δ must be of greater power than σ and therefore than ρ : τ would then be of higher power than all right facing points in the interval α to γ and also in the interval γ to δ , and therefore would have been the original position of β .

Juxtaposition and omitted points. In most systems there is some operation which is described simply by juxtaposition, without any special operator. In Church's system this is the application of a function to its argument; in Russell's it is conjunction and in algebra it is multiplication. In such systems the abbreviated formulae will be less restricted than the abbreviated formulae in the sense defined here. It is also usual to omit some of the points in the abbreviated formulae, it being understood that a point is to be introduced wherever one is necessary in order to satisfy the conditions (a), (b), (c), above. The power of such points may be settled at the same time as the other power conventions. There is one matter which has been left doubtful about the introduction of these points. When a pair of brackets is adjacent to operators at each end one of the brackets must be 'protected' from its operator by a point, but only one, in order to satisfy (b); which bracket should it be? The following three rules are equivalent:

(1) One may put in a point in both places. In this case (b) is no longer satisfied, and the final result of removing the points, by either of the rules, leaves an otiose pair of brackets which have to be removed before we have an unabridged formula.

(2) Both points are put in and then the weaker one removed.

(3) If the conventions below are adopted one may put the point in after the brackets.

With this practical kind of system, where juxtaposition is used and some points are omitted, the abbreviated formulae do not satisfy the conditions (b), (c) above: they satisfy (a), however, and also (c') below. To distinguish these formulae from the abbreviated formulae proper I will call them *practical formulae*. The conditions (a), (c') are necessary and sufficient for being a practical formula.

(c') No pair of consecutive symbols in the projected formula may be one of the following: '(), '(:', '(D', ':)', '::', 'M)', 'MD', 'D)', 'DD'. No three consecutive symbols may be 'M:D' or 'D:D'. The projected formula may begin only with '(', 'x', or 'M' and may end only with ')' or 'x'.

From a practical formula we can obtain an abbreviated formula by first introducing an operator \star to take the place of juxtaposition, and afterwards the omitted points. Wherever a point π is not adjacent to an operator we replace it by ' $\pi\star\pi$ '. We replace ')' by ') \star (', 'A' by ') \star A', 'A(' by A \star (' and

$'AB'$ by $'A \star B'$ if A and B are irreducible formulae. We then replace the omitted points. We may use small circles to represent them: thus the sequences ' xD ', ' Dx ', ' Mx ', ' MM ', ' DM ', ' xM ', ' M ' in a projected formula become ' x_0D ', ' D_0x ', ' M_0x ', ' M_0M ', ' D_0M ', ' x_0M ', ' M '. The last two of these must be again modified by the introduction of \star , giving ' $x_0 \star_0 M$ ' and ' $M \star_0 M$ ' but the process then comes to an end.

Application to Church's system. In Church's system the irreducible formulae are the variables and other single letter formulae, including, if we wish, abbreviations such as S_{\dots} . The monadic operators are \sim , $[x_\alpha]$, $[\exists x_\alpha]$, $[\forall x_\alpha]$, λx_α , and $[\lambda x_\alpha]$, of which the last two may be regarded as the same so far as the unabbreviated formulae are concerned. The dyadic operators are \supset , \mathbf{v} , \equiv , $\&$, $=$, \neq , to which we may add \star . If we adopt the conventions of the last section it is only necessary to decide on the relative powers of the points in order that the unabbreviated form of a practical formula should be determined. The conventions recommended are as follows:

We divide the operators into two classes:

Class of high power containing \supset , \mathbf{v} , $\&$, \equiv , \sim , $[x_\alpha]$, $[\exists x_\alpha]$, $[\forall x_\alpha]$, $=$, \neq , and others which may be added from time to time such as $>$, $<$, $/$.

Class of low power containing λx_α , \star , and others which may be added from time to time such as $+$, $-$.

In the class of high power we distinguish some operators as *handicapped*: these are $=$, \neq (and $>$, $<$). A point adjacent to an operator in the high power class is always of higher power than one in the low power class. In the case of two points adjacent to operators of the same class the one with the greater number of dots is of the higher power, with the provisos that if the operator is handicapped the number of dots must be reduced by one, and that a point which is either omitted or represented by \circ counts as of 'zero dots.' Amongst points of the same class, and having the same (corrected) number of dots the left facing points are of higher power than the right facing. There is no need to decide which shall be the more powerful of two similarly facing points, since this is irrelevant to the scope condition, but for definiteness let us say that the one on the left is the more powerful.

The 'unabbreviated formula' which results from a 'practical formula' by the application of one of our rules is not strictly speaking a formula of Church's system nor even an abbreviation of one which would be recognised by Church. If A is the unabbreviated formula, and $A^{(D)}$ the corresponding formula recognised by Church, then $A^{(D)}$ is A if it is an irreducible formula, and otherwise is defined inductively by the conditions that:

$$((A) \star (B))^{(D)} \text{ is } (A^{(D)} B^{(D)}),$$

$$((A) \supset (B))^{(D)} \text{ is } [A^{(D)} \supset B^{(D)}],$$

$$((A) \mathbf{v} (B))^{(D)} \text{ is } [A^{(D)} \mathbf{v} B^{(D)}],$$

etc.;

$$\begin{aligned}
 (\sim(A))^{(D)} &\text{ is } [\sim A^{(D)}], \\
 ([x_\alpha](A))^{(D)} &\text{ is } [(x_\alpha) A^{(D)}], \\
 &\text{etc.}; \\
 ([\lambda x_\alpha](A))^{(D)} &\text{ is } (\lambda x_\alpha A^{(D)}), \\
 (\lambda x_\alpha(A))^{(D)} &\text{ is } (\lambda x_\alpha A^{(D)}).
 \end{aligned}$$

Discussion of the conventions. These power conventions appear to differ markedly from the Russell conventions because the operator against which a point is placed is made to be of greater effect in determining the power than the number of dots. However in Russell's system the operators in our class of low power do not occur at all, and the difference must be thought of as a rejection of his distinctions between operators for punctuation purposes, together with a special new treatment of the new operators. Our 'handicap of one dot' convention for $=$, $>$, etc. may however be regarded as taking the place of some of Russell's distinctions.

It is easy to remember which are the operators in the class of high power. They are the ones which normally either operate on propositions or form propositions. The ones which are handicapped are those which form propositions but do not normally operate on propositions. The case of $[\lambda x_\alpha]$ is exceptional, but again it is easy to remember its power because the notation has been made analogous to that of the other high power operators. One would not normally use the form $[\lambda x_\alpha]$ unless it is operating on a proposition.

The reason for adopting our high and low power class conventions is that in practice it is extremely seldom that we want the scope of a bracket starting from one of the low power operators to include one of the high power operators. The low power operators are in fact just the ones that we should use in formalising the mathematical formulae in a mathematical book. We should use the high power operators in formalising the English connecting matter. It is hardly necessary to point out that a bracket in one of the formulae never pairs with one in another formula, with English intervening. Our convention has the desired effect of closing automatically all brackets outstanding in the 'mathematical formulae' before going on to the English text. The reasons for adopting the handicap convention are similar. A bracket starting from an equality sign will not usually enclose another high power operator, although a bracket from an operator of low power will not enclose an equality sign.

The convention by which left facing points are made more powerful than right facing is convenient to complete the conventions, and is also in agreement with two of Church's own conventions, viz. that in the absence of other indication association is to the left, and that in the absence of dots an omitted bracket has the minimum possible scope.

The use of square brackets in connection with some of the operators, e.g. $[\exists x_\alpha]$, is necessary in a theoretical treatment, but it is not suggested that such a notation should be generally adopted. With very few exceptions one can tell

whether the round brackets are part of an operator or not. One exception is the formula $(p_{\circ\circ})(q_{\circ})$.

Examples. (i) As a first example of the effects of our conventions I shall take a very simple formula and remove the dots by the first rule. The formula which I shall take is $ab.c$ and even this will be found quite sufficiently complicated for the purpose. We must first transform the 'practical formula' into an 'abbreviated formula' by introducing the operator \star , and the points \circ . This gives us $a_{\circ}\star_{\circ}b\star_{\circ}c$. We now take the point of highest power, which is the one following the b and replace it by a bracket facing left, i.e., away from its operator, and balance it with a bracket at the left end, giving us $(a_{\circ}\star_{\circ}b)\star_{\circ}c$. We now have to evaluate separately $a_{\circ}\star_{\circ}b$ and $\xi\star_{\circ}c$. The stronger point in $a_{\circ}\star_{\circ}b$ is the left one and this formula is therefore equivalent to $(a)\star_{\circ}b$, i.e., to the result of substituting (a) for η in the unabbreviated form of $\eta\star_{\circ}b$, i.e., in $\eta\star(b)$. The unabbreviated form of $(a_{\circ}\star_{\circ}b)$ is therefore $((a)\star(b))$: also the unabbreviated form of $\xi\star_{\circ}c$ is $\xi\star(c)$, and therefore the unabbreviated form of $(a_{\circ}\star_{\circ}b)\star_{\circ}c$ is the result of substituting $((a)\star(b))$ for ξ in $\xi\star(c)$, i.e., is $((a)\star(b))\star(c)$. Transforming this back to a formula of Church's system, properly speaking, we get $((ab)c)$.

In the remaining examples we will always use the second rule. No type suffixes will be shown.

(ii) We will first deal with formulae without operators, or at least without operators of high power. As one example,

$$(a((cd)(e(fg)))))$$

can be abbreviated to

$$a\mathbin{::} cd\mathbin{::} e\mathbin{::} fg.$$

As another,

$$a\mathbin{::} cd\mathbin{::} e\mathbin{::} fg$$

is an abbreviation of

$$((a(cd))((ef)g)).$$

The association to the left rule has been used here: in other words we have had to apply the rule that a dot is more powerful in its left facing than its right facing aspect. The structure of a formula is often more easily taken in if we slightly increase the number of dots and do not rely on this rule, e.g. the same formula may be written

$$a\mathbin{::} cd\mathbin{::} e\mathbin{::} fg,$$

or again as

$$a\mathbin{::} cd\mathbin{::} ef\mathbin{::} g.$$

Similarly it is often not advisable to replace all of the brackets in a formula by dots. As a group of dots never replaces more than four brackets it can hardly ever be worth while having as many as six dots, say, in a group. A few dots can however be made to go a long way by mixing them judiciously with explicitly shown brackets, e.g.

$$bc.d::e::f:g:h.ij$$

is the best form of a certain formula when expressed without any explicit brackets, but

$$bc.d::e.f(g:h.ij)$$

is a much better form of it.

As an example of a formula involving λ ,

$$h::\lambda f\lambda x.fx:g$$

is an abbreviation of

$$(h((\lambda f(\lambda x(fx)))g)).$$

(iii) As an example of a more general type of formula,

$$[m].Nm \supset [p].Np \supset m \neq S:pS.m$$

is an abbreviation of

$$[m]((Nm) \supset ([p]((Np) \supset (m \neq S((pS)m))))).$$

If we did not have the 'handicap of one dot' convention we should have to put in a dot after ' $Np \supset$ '. In this case the effect is slight, but sometimes it can be considerable, e.g. without the convention

$$[x].x=y \& y=z \supset x=z$$

would have to become

$$[x]:x=y \& .y=z \supset .x=z.$$

(iv) The expressions

$$p \supset .q \supset :r \vee s . \& t : \& u$$

and

$$p \supset (q \supset ((r \vee s) \& t)) \& u$$

and

$$p \supset (q \supset .r \vee s \& t) \& u$$

are all abbreviations of the same formula. Notice that in the first of these expressions the bracket starting after ' $p \supset$ ' does not close when we reach the stronger point on the left of ' $\& t$ ', because the former is reinforced by the even stronger point after ' $q \supset$ '. The most legible form of this formula, if it is standing by itself, is probably

$$p \supset :q \supset :r \vee s . \& t : \& u.$$

(v) A formula similar to the last example in one respect is

$$p \supset q \& r,$$

which with our conventions is an abbreviation of

$$(p \supset q) \ \& \ r,$$

but with Russell's or Church's conventions would be an abbreviation of

$$p \supset (q \ \& \ r)$$

on account of the subdivision of our 'class of high power' into smaller classes of different powers.

(vi) Normally we shall not want to put dots against equality signs, or other operators which form propositions but do not operate on propositions. A typical exception is

$$[i x_\alpha] \cdot g_{\alpha} x_\alpha \supset f_{\alpha} x_\alpha : = y_\alpha.$$

Another type of freak formula, difficult to abbreviate, occurs when we have functions which take propositions as arguments, e.g.

$$h_{\alpha\circ}(p_\circ \supset q_\circ).$$

The only way of avoiding explicit brackets in such a case is to express the implication, not with the implication operator but with the implication function $C_{\circ\circ\circ}$, thus

$$h_{\alpha\circ} \cdot C_{\circ\circ\circ} p_\circ q_\circ.$$

KING'S COLLEGE, CAMBRIDGE

Practical Forms of Type Theory

(J. Symbolic Logic, vol. 13 (1948), pp. 80–94)

PREFACE

The first draft of this paper (*A Practical Form of type theory*) shows that it was originally intended as one of several. One aim, which became the main aim, is set out in *Reform*. Turing wished to encourage ‘mathematicians-in-the-street’ to use notation and forms of argument which would safeguard their work from ambiguity and inconsistency; but to do this without forcing their work into the straitjacket of a particular logical system, or even requiring them to have detailed knowledge of such a system. To the end of his life, he thought this aim a proper one for a logician, and from time to time gave talks to mathematicians in which he would expound particular logical points. As a logician, however, he was interested in devising formal systems which could act as bridges between the formal and the informal, and this motivated him to produce the two systems set out in this paper. In S1, besides describing the intended universe of the nested-type system, he also explains a number of elementary logical points. He did not expect mathematicians to use the system, but it looks as if he hoped that some mathematicians would read the paper, even though ignorant of symbolic logic. In this, as in some of his other papers and lectures, he was overly optimistic about the abilities of his intended audience. Not only is it not obvious that the rules and axioms do correctly formalise the informal notions, but, more explicitly, a reader unfamiliar with symbolic logic will not appreciate the vital distinction between mathematical and metamathematical statements. (When in 1948 Turing tried to explain the Deduction Theorem to me, I failed to understand it because I did not distinguish between ‘ B can be inferred from A ’ and ‘ $A \supset B$ ’.)

Work on the nested-type theory, including the writing of *A Practical Form of type theory* had mostly been done before the summer of 1945, when Turing moved from Hanslope Park to the National Physical Laboratory. There, during the second half of 1945, he was fully occupied working out his proposals for the ACE computer (*The Collected Works, Mechanical Intelligence*, pp. 1–86). During 1946 he completed *Practical Forms*. I do not know whether he merely shelved or completely abandoned further work on *Reform* and the project described in it.

Notes and Corrections

1. p. 85, third paragraph.

Replace ‘(rules I–X, XI_n)’ by ‘(rules I–VIII, IX_n)’

2. p. 85.

In the definition of $(\exists!x, r)P$, replace ' \neq ' by '&'

3. pp. 86–7.

Replace 'X' by 'x'.

4. p. 86, line b4.

Replace ' $T \supset T$ is F ' by ' $T \supset T$ is T '.

5. p. 88, (4).

The 'formal justification' referred to is given by Theorem A of PFT II.

6. p. 89, first paragraph.

The stated equivalence is Theorem B of PFT II.

7. p. 93, Rule III.

Add 'provided x is not free in H '.

8. p. 94, Rule VII.

Replace ' x, y, z, u ' by ' y, z, u '.

The nested-type system

It is distinguished from other well-known formalisations of type-theory in three ways.

- (a) The types are cumulative: each type is included in all higher types.
- (b) Type restrictions on variables are only introduced by the variable-binding operator $(\lambda x, r)$ and (x, r) ; and an application term (AB) is well-formed irrespective of the types that may be assigned to the terms A and B . Thus xx and $x((\lambda y, r)x)$ are well-formed.
- (c) To avoid paradox an element C , of type 0, (denoting undefinedness) is introduced and is the value of any term which transgresses the doctrine of types. In conversation, Turing called C the 'nonsense element'; because the types are cumulative, only one such element is required. However, the clause ' $\sim D^0y$ ' in Rule VII implies that a totally undefined function such as $(\lambda x, 0)C$ is not of type 0 and so is different from C . Then, by the axiom of extensionality, $(\lambda x, r)C = (\lambda x, 0)C$ for any r , so that there is only one totally undefined function. (The reviewer in the JSL, not realising this, claimed the system to be inconsistent, but later published a correction to his review.) Nowadays, in recursion theory and the theory of computation it is quite usual to admit an undefined (or, in lattice-theoretic terms, a bottom) element. As in Turing's system, this allows partial functions to be interpreted as total, but it should be noted that in Turing's system, propositions are either true or false; $xx = C$ is true whilst $xx = T'$ is false.

It may help the reader to realise that the assignment of types, as described in the informal description, is effected by Rule VII and the axioms A2–A5 in the following way. Let f be a function which somewhere takes a value different from

C (e.g. $(\lambda x, r)((\lambda x, 0)C)$). Then by Rule VII and axiom A3 the least type of f is one greater than the least type of any argument x for which $fx \neq C$; and by A5, if, for some y , fy is not of type 0 then the least type of f is one greater than the least type of any value taken by f .

The Equivalence Theorem

This states that the nested-type system is equivalent to Church's system – see p. 90 of Turing's paper. When two formal systems share the same logical apparatus, then they are equivalent in Turing's sense if and only if they are mutually interpretable; and the translation will give an intelligible (perhaps unintended) meaning in one system to the non-logical primitives of the other. The nested-type system does not share a common logical apparatus with Church's system and so Turing's definition is weak; the translations need not, in any sense, preserve meaning. I think that Turing added the conditions (v) and (vi) (on p. 89) solely to ensure the transitivity of equivalence. More sensible conditions would be, e.g.:

(v') If A and B are proposition-like formulas of 1 then in 2 we can prove

$$(A \supset B)^{(1,2)} \equiv (A^{(1,2)} \supset B^{(1,2)}),$$

and similarly for (vi'). These would ensure that translation preserved inferences and propositional combinations. In fact, the actual translations given by Turing in *Practical Forms II* do preserve meaning as well as satisfying (i)–(vi).

Noun-Classes

Turing was pleased with this idea (to be found on p. 92) as showing that natural language usually conforms with a doctrine of types. In *A Practical Form of type theory* the claim is made more vividly than in *Practical Forms*

... one tends to feel that Russell's type theory was largely anticipated by prehistoric man.

The idea, and its origin, is further discussed in *Reform*. For mathematics, the following inductive definition represents Turing's line of thought, but is not given explicitly by him.

- (A) The ground type $\{x \mid D^0x\}$ is a noun-class.
- (B) If A and B are noun-classes then so are

$$Pot(A, B) (= \{f \mid f : B \rightarrow A\}) \quad \text{and} \quad A \cup B$$

- (C) A formula (or term) is *proper* if each bound variable which occurs in it is restricted to some noun-class.

- (D) If A is a noun-class and $\phi(x)$ is a proper formula in which x is the only free variable, then $\{x \mid x \in A \& \phi(x)\}$ is a noun-class.

This definition has the following consequences:

- (1) For each r , $\{x \mid D^r x\}$ is a noun-class.
- (2) Every formula of the nested-type system is equivalent to a proper formula.
- (3) If A is a noun-class then for some r , $A \subseteq \{x \mid D^r x\}$. (In both *Practical Forms* and *A Practical Form of type theory* this is taken as the definition of noun-class.)

REMARK. If one introduces, or defines, the set \mathbf{N} of natural numbers (equipped with zero and successor) as a noun-class, then careful examination of the standard definitions of concrete (as opposed to abstract or axiomatic) analysis shows that all the concepts involved (such as \mathbf{R} , continuous real functions, \mathbf{C}^∞ , \mathbf{L}^2 , the space of bounded linear operators from \mathbf{L}^2 to \mathbf{L}^2) are noun-classes, and the theorems about them can be represented by proper formulae; the variables which occur are, in fact, always restricted either by hypothesis or by typographical convention. This, together with (2) and (3) ensures that everything can be expressed in the nested-type system. Thus the instructions given by the logician to the mathematician can be summed up by: check that all propositions which occur can be expressed by proper formulae. It seems to me that this is an easier and more natural way to proceed than the use of the concealed-type system. Perhaps this is, more or less, the other system which the ‘author has investigated’.

Concealed-type systems

The mathematician may wish to use statements such as $(x)(x = x)$ which are true for unrestricted variables – for ‘anything whatever’. Some of these are allowed by the (different) concealed-type systems of *Practical Forms* and *A Practical Form of type theory*. These systems depend on a notion of interpretability; it is convenient to treat both together, and to give definitions which are more transparent than those given by Turing.

Let x_i ($i = 1, 2, \dots, n$) be the bound variables of a term or formula A ; suppose that lower type-bounds r_i^0 have been assigned to them. A *typed version* of A is obtained by restricting each x_i to $D^{r_i^0}$ for some $r_i \geq r_i^0$. The formula (or term) A is *interpretable* if values for the r_i^0 can be found such that any two typed versions of A are provably equivalent (or equal). In *A Practical Form of type theory* it is required that r_i^0 can be determined outright; in this case, I shall use the term ‘strongly interpretable’. In *Practical Forms* each r_i^0 may depend on the types assigned to the free variables of A and on the types which have been assigned to those bound variables within whose scope the binding occurrence of x occurs.

The difference between the two notions is illustrated by

$$(x)(\exists y)(yx = T'). \quad (*)$$

This is not strongly interpretable, but it is interpretable, because

$$(x, r)(\exists y, s)(yx = T')$$

is provable if $r \geq 0$ and $s \geq r + 1$.

In *A Practical Form of type theory* Turing remarks that every proper formula is strongly interpretable; he sketches a concealed-type system which is like that in *Practical Forms* except that ‘admissible’ is replaced by ‘strongly interpretable’. As he says in *Practical Forms*, the metamathematical business of showing that a given term or formula is interpretable is likely to be heavy-going. So he introduces the notion of admissibility. This may lighten the formal work, but it is counter-intuitive. For example, $(\exists x)(x = x)$ and $(*)$ are not admissible formulae, and cannot be proved in the system. Of course, every proper formula is admissible; for a non-trivial admissible (and provable) formula whose meaning cannot be expressed by a proper formula, consider

$$(x)(\exists y)(Pot(x, x)y = T').$$

Unlike the use of noun-classes and proper formulae, the concealed-type system of *Practical Forms* seems artificial, and not user-friendly. In the 1950’s, Turing and I discussed Church’s system and the nested-type system, but I do not recall his ever referring to the concealed-type system.

The first draft (A Practical Form of type theory)

The first two pages give a fuller account of Turing’s motivation than does *Practical Forms*, so I quote them in full.

It is usual for mathematicians to pay lip-service to the theory of types, but they will not usually make any attempt to bring their mathematics into line with it. An occasional paradox may perhaps be attributed to neglect of types, but no suggestions are made for the avoidance of these paradoxes short of the expression of all mathematics in the formalism of *Principia Mathematica* (say). In the present paper a system will be described which takes account of type theory, but at the same time follows very closely the normal mathematical outlook. The type theory intrudes itself on the system only very slightly, and its effect may be summed up in the form of one or two simple and natural cautions, which are easily carried over to unformalised mathematics: this should, I hope, enable all such serious mathematics as is supposedly based on

the theory of types to be brought genuinely into line with it, at the cost of very little additional trouble to mathematicians.

This paper will appear in three two parts. The first part is written chiefly for the mathematician who wishes to increase the rigour of his proofs along the lines indicated in the previous paragraph, rather than for the logician. The emphasis will be on notation and meaning rather than on axioms and rules of procedure; these will, however, be given for the sake of completeness. The second part will be devoted to a little axiomatic development, and the justification of the system in the case of the 'finite universe' i.e. the case where there is only a finite number of individuals. It will establish a very complete connection between this system and that of Church. This connection seems to be valuable because Church's system has greater theoretical simplicity than the proposed 'practical system', but is less convenient for the formalisation of proofs. Consequently, it will be natural to express proofs in the practical system, but metamathematical results in terms of Church's system.

The author wishes to repudiate any implication that may be suggested by this paper to the effect that he believes the Russell philosophy of mathematics to be the truest. He does believe, however, that it is the one which is most easily understood, and also that it describes most closely the accepted form of present-day mathematical thinking. This paper is concerned with giving accurate expression to that thinking. When that is done it will be easier to see the limitations of the outlook which goes with this form of thinking.

The description of the nested-type system is much as in *Practical Forms*, except that the deduction theorem is taken as a primitive rule of inference. After two pages describing the use of hypotheses in mathematical argument, and explaining that the free variables in a hypothesis can be regarded as constants, Turing writes:

In the present system we have taken the bull by the horns and adopted the deduction theorem and the whole technique of hypotheses as fundamental.

I have already described the concealed type-system of *A Practical Form of type theory*. At the end of it, Turing writes:

It would be natural at this point to illustrate the ideas involved by examples of well-known mathematical results expressed in the manner recommended above. There are two reasons why this is not being done. In the first place, as has been mentioned, it is not considered desirable to convert mathematics into a branch of symbolic logic: examples of the kind would therefore consist of long passages of very conventional nature, with occasional very slight variations from the conventional pattern. Secondly, it is the author's intention shortly to make a very comprehensive review of current mathematical notation, which will involve the discussion of such examples. In these discussions there will be many other points of interest, but the type aspect will also be included: the present moment is therefore thought premature for the consideration of examples.

In the last section, he states (as ‘Theorem A’) the completeness of the rules and axioms for the nested-type system with the axiom IX_N which expresses that there are just N individuals; and ‘Theorem B’ is the equivalence theorem referred to in the last paragraph of S3 of *Practical Forms*.

PRACTICAL FORMS OF TYPE THEORY

A. M. TURING

Russell's theory of types,¹ though probably not providing the soundest possible foundation for mathematics, follows closely the outlook of most mathematicians. The present paper is an attempt to present the theory of types in forms in which the types themselves only play a rather small part, as they do in ordinary mathematical argument. Two logical systems are described (called the "nested-type" and "concealed-type" systems). It is hoped that the ideas involved in these systems may help mathematicians to observe type theory in proofs as well as in doctrine. It will not be necessary to adopt a formal logical notation to do so.

1. The nested-type system for a finite universe. In this section the notation of the nested-type system will be explained. The explanation will be in terms of the 'finite universe,' i.e. we start with a finite number of objects or 'individuals' and build up other entities from these. We can then formulate certain rules which give valid results in this case and hope that they will apply in the infinite case also. We cannot of course hope that all such rules will work. We have to imagine that many rules of this kind have been tried, found wanting and rejected, and that others are still in use. This rather unsatisfactory-sounding process is as good an account as the author feels can be given of the way in which current mathematical procedure has grown up. But whatever the truth of this may be the finite universe provides a first class ground on which to describe the nested-type system, and we proceed accordingly.

Our finite universe has initially as its members the 'individuals' U_1, \dots, U_N . Although these include all the individuals, they need not exhaust our stock-in-trade, for we can also bring in functions taking the individuals as arguments and having them also as values. With our increased range of commodities we can then go into business again and produce a still greater variety of objects, and repeat without limit. There obviously arises a great variety of different kinds of functions which may need to be distinguished, but for the present system we need only trouble ourselves with the very broadest divisions, which will be called types. These divisions are described below.

The individuals U_1, \dots, U_N form type 0.

The functions of individuals, taking individuals as values, together with the individuals themselves, form type 1.

The functions of arguments in type 1, taking values also in type 1, together with the members of type 1, form type 2.

.....

The functions of arguments in type n , taking values also in type n , together with the members of type n , form type $n + 1$.

.....

Received January 6, 1947.

¹ A. N. Whitehead and Bertrand Russell, *Principia mathematica*, Cambridge, England, 1925.

It must be understood that by a "function" we mean the function itself and not merely one of its values. To illustrate the point by analogy with functions of a real variable, we should say that "sin" denotes a function, but that "sin 0.3" and "sin x " do not, although the latter is often used (incorrectly in the author's opinion) as if synonymous with "sin".

It is convenient to require functions to be defined throughout the appropriate type, i.e. not to permit such definitions as " $f(0) = 0$, but if x is different from 0 then $f(x)$ is undefined." In order to cover such cases we shall set apart from the outset a particular individual U_1 , which we shall rename " C ", to be the value of a function in all cases where it would normally be regarded as undefined. So far as possible we try to keep C on a par with the other individuals. We deviate from this principle by adopting the convention that the value of a function is always C unless the function is of higher type than the argument. (More strictly, if the function belongs to every type to which the argument belongs.) We respect the principle by refraining from considering every expression containing " C " to have the value C .

The functions and individuals together will be known as *terms*. With our finite universe it is convenient to think of the functions as given by tables, consisting of two columns, in the first of which appear all the necessary arguments, and opposite them in the second column the appropriate values. Thus with $N = 4$ a typical member of type 1 would be represented by the table

(1)

U_2	U_3
U_1	U_1
U_3	U_1
U_4	U_4

It would be a convenience to have the table rearranged with the first column in natural order. In the case of the above table (1) we should simply have to interchange the first two rows. Such a table may be said to be in normal form. We can do this for all tables of type 1, and when we have done so we are in a position to define a natural order for the members of type 1. With both tables in normal form, the earlier table is to be the one which has the earlier value in the last row in which the two tables differ. Thus the table (1) above precedes

(2)

U_1	U_1
U_2	U_4
U_3	U_3
U_4	U_4

since when (1) is put into normal form the two tables differ last in the third row, and there (1) has the value U_1 but (2) has the value U_3 . We shall also adopt the convention that the individuals in type 1 precede the tables. We may now continue the numbering of terms so as to include all type 1, simply numbering them in the natural order just defined. The numbers will extend from 1 to $N + N^N$. It may be verified that the above tables (1) and (2) are U_{205} and U_{241} respectively. A similar process may now be carried out for type 2 and then for type 3. In

general when we are dealing with type n we have already numbered the members of type $n - 1$. It is easily verified that those tables which have already appeared as members of type $n - 1$ have the order which they had in that type, and precede all the new tables. The order of any two tables (new or old) is that of the last pair of values in which they differ.

Let us now introduce the notation (UV) to denote the result of looking up V in the table U ; in slightly different words it is the entry against V in the table U .² In other words again it is the value of the function U for the argument V , and might therefore, in agreement with current mathematical practice have been denoted by $U(V)$. Our conventions require (UV) to be C in cases where the table gives no information: these are just the cases where the lowest type to which U belongs does not exceed the lowest for V . We may also introduce the notation $U = V$ to denote the identity of the terms U and V . It should be noticed that so long as U and V are tables known to belong to some particular type n we can establish their identity by showing that they have the same values throughout type $n - 1$ (this is known as the principle of extensionality and gives rise to the "axiom of extensionality"). The principle fails for individuals, for if U and V are individuals then (UX) is always identical with (VX) , both being C , and yet U and V may well be different. The principle also fails when the types of the terms are unknown, for we can never then be sure that we have examined sufficient arguments for the functions. There may be some argument in a higher type than we have yet considered for which the two functions differ.

The expression $U = V$ which we have just introduced denotes a *proposition*, unlike (UV) which was a term. Propositions may be thought of as having a value which is either truth (T) or falsity (F). By taking T and F to be individuals we could have arranged for the propositions to be included amongst the terms, but we have not in fact done so.

There are several other ways of forming propositions. If P and Q are propositions then $(\sim P)$ is a proposition whose value is opposite to that of P and $(P \supset Q)$ is one whose value is F if and only if P is T and Q is F. We may read $(\sim P)$ as "not P " and $(P \supset Q)$ as " P implies Q ." If U is a term then $D^r U$ represents the proposition that U is in type r , i.e. it is T if and only if U is in type r .

We could of course introduce a great variety of further means for forming terms and propositions. We could for instance define $(P \& Q)$ as a proposition whose value is T if and only if both P and Q are T. We shall be content however with comparatively few, namely those we have already introduced, together with one further way of forming propositions and one of forming terms. These cannot be described without bringing in the ideas of "variable" and "formula with variables." Variables are of little importance except as parts of formulas. All we need say about them is that as a matter of notation small italic letters with any number of primes will be used as variables. *The letters p, q, r, s, t ,*

² We shall use heavy type letters throughout to represent variables or undetermined formulas or tables. They occur only in metamathematical discussions. All our statements are understood to be true whatever substitutions of formulas (or tables, as the case may be) are made for the heavy type capital letters, and whatever substitutions of variables are made for the small heavy type letters.

(possibly with primes) will be proposition variables and the others term variables. Small heavy type letters may be used to stand for any variable, with an obvious convention concerning the kind of variable. An example of a "formula with variables" is the expression $x = U_5$. On substituting a term, e.g. U_{10} for the term variable x it becomes a proposition. Similarly $(U_{405} x)$ is a formula with variables: in this case substitution yields a term. In general a formula with variables or more briefly a *formula* is an expression which yields a term or proposition on substituting terms and propositions for the (free) term and proposition variables respectively. The formulas may be called *term formulas* or *proposition formulas* according as they give rise to terms or propositions on substitution. The word *free* in the definition should be ignored for the present.

We can now describe our remaining ways of forming terms and propositions. If P is a proposition formula with only the one free term variable x and no proposition variables then $(\mathbf{x}, r) P$ is a term and $(x, r) P$ is a proposition. Of these the term $(\mathbf{x}, r) P$ has the value C unless there is one and only one term U in type r for which the result $S_U^x P$ of substituting U for x in P is T : if there is a unique U with this property then the value of $(\mathbf{x}, r) P$ is that U . The value of the proposition $(x, r) P$ is T if and only if all the results of substitution, $S_U^x P$, with U in type r , have the value T . We may read $(\mathbf{x}, r) P$ as "the x in type r such that P " and $(x, r) P$ as " P , for all x in type r ."

Now consider the expression $(x, 3)(x = y)$. In it there occur the two variables x and y . If we substitute a term, e.g. U_6 , for y we shall obtain a proposition, but if at the same time we substitute U_9 for x we shall obtain nonsense. We would like to excuse ourselves from making this second substitution and admit $(x, 3)(x = y)$ to membership of the class of formulas. Our excuse is that substitution should only be made for the *free* occurrences of a variable, and that the occurrences of x in $(x, 3)(x = y)$ are not free but bound. We say that a variable u occurs *bound* in a formula if the occurrence in question is in a part of form $(u, r) P$ or $(u, r) P$. Thus the first occurrence of x in $(y, 1) [x = (\mathbf{x}, 0) (x = x)]$ is free and the others are bound. This expression is a proposition formula according to our definition. To verify this, first note that $x = x$ is a proposition formula with no free variables other than x and that $(\mathbf{x}, 0) (x = x)$ is therefore a term. Consequently $U = (\mathbf{x}, 0) (x = x)$ is a proposition, and *a fortiori* a proposition formula, for any term U . It has no free variables other than y (indeed it has none at all), and therefore $(y, 1) [U = (\mathbf{x}, 0) (x = x)]$ must be a proposition for any term U , i.e. $(y, 1) [x = (\mathbf{x}, 0) (x = x)]$ is a proposition formula.

It will now be seen that terms and propositions are just term formulas and proposition formulas without free variables.

Free and bound variables are familiar in mathematics though they are seldom consciously recognized. A typical example of a bound variable is that of x in the integral $\int_0^1 x dx$; x occurs free in the equation $x(x - 1) = 0$. A convenient method of distinguishing between bound and free variables is to make a substitution of a constant (of the appropriate kind) for the variable in question. If nonsense results the variable is certainly bound: if sense results it is most probably free. Sense may perhaps result from substitution for a bound variable

if the result of the substitution and the original expression are interpreted according to different conventions. The double suffix summation convention of tensor theory provides an example of this. Using this convention the variable j in the expression $a_{i,j}b_{jk}$ is bound, but we can substitute 1 for j and obtain a perfectly sensible expression; it is sensible because it is interpreted without applying the double suffix convention.

The outcome of our definition of "formulas" is that they will include terms, propositions, and variables. Also if A and B are term formulas, P and Q proposition formulas, x a term variable, and r a numeral representing a non-negative integer, then $(A B)$ and $(,x, r)P$ are also term formulas and $(A = B)$, $D^r A$, $(\sim P)$, $(P \supset Q)$, and $(x, r)P$ are proposition formulas. Our use of the letter " r " in these cases must not of course be confused with its use as a proposition variable. One further method of constructing formulas is worth mentioning although it is possible to do without it, and define it in terms already explained. This is "abstraction." If A is a term formula then $(\lambda x, r)A$ is a term formula of type $r + 1$. It stands for the function whose value for the argument U in type r is $S_U^* A$, provided that $S_U^* A$ is in type r for every U in type r : if however there is a single argument U in type r for which $S_U^* A$ is not in type r then $(\lambda x, r)A$ is C . We can define $(\lambda x, r)A$ in previously explained terms as

$$(\lambda y, r + 1) (\sim[(x, r) (yx = A) \supset D^0 y])$$

where y is any variable not occurring free in A .

In the case of a finite universe the individuals U_1, \dots, U_N form a part of the system. When dealing with an infinite universe this does not seem to be necessary, but it is convenient to retain symbols for three of them; these are U_1 which is called C and which we have already mentioned, U_2 which is called T' and U_3 which is called F' . These last two may be regarded as unofficial representatives of truth and falsity, looking after their interests amongst the terms: their official representatives are T and F which are propositions. The chief use of T' and F' is in connection with propositional functions. If we wish to express 'x is mortal' we form a function M which is defined for individuals (supposed to include mammals) and has the value T' for mortal arguments, F' for immortal arguments. Then "x is mortal" is written as $Mx = T'$.

At this point we should pause and consider what we have done. We have defined a class of expressions which we have called term-formulas and proposition-formulas, and which roughly correspond to the terms and propositions of mathematics. These formulas are given interpretations in the finite universe in terms of individuals and tables. Each term formula without free variables has an interpretation as a particular individual or table, and each proposition formula has an interpretation which is truth or falsity. We are able to determine whether a proposition formula without free variables is true by working out its interpretation, although this will be a very lengthy business unless the formula is very simple and N very small. The work involved in establishing the truth of formulas can be greatly reduced by the use of various rules, e.g. that if two formulas P, Q are true then $\sim(P \supset \sim Q)$ is true. A process of application of such rules may be allowed to oust the process of working out the interpretation.

Since the majority of the rules involved do not make any reference to the number N it is easy to forget the finite universe, and to allow the various rules to become reflex action. Eventually we break off almost all connection with the finite universe picture: in particular we repudiate such propositions as

$$(x, r)(y, r)((x \neq y) \supset ((fx) \neq (fy))) \supset (x, r) (\exists y, r)((fy) = x)$$

which are especially connected with such a picture. Finally we even repudiate the picture more violently by adopting an "axiom of infinity."

This, in my opinion, is a very idealised but essentially correct account of how the present mathematical argument-form has grown up. The last step or two may appear very lame, but I think this cannot be helped: I think that these last steps are not really sound.

One set of rules which can replace the finite universe picture is given below in §2 (rules I–X, XI_n).

ABBREVIATIONS. At this point we are obliged to introduce a few conventions which permit us to abbreviate our formulas. The unabbreviated formulas would be disagreeably cumbrous.

(a) We may introduce abbreviations by means of the arrow: a formula standing to the left of an arrow is understood to be an abbreviation of that on the right of it. If heavy type letters appear in these expressions it is understood that the formula on the left is an abbreviation of that on the right for any meaningful substitutions of formulas for the heavy-type letters. With these conventions we introduce the abbreviations:

$$\begin{aligned} (P \ \& \ Q) &\rightarrow (\sim(P \supset (\sim Q))) \\ (P \vee Q) &\rightarrow ((\sim P) \supset Q) \\ (P \equiv Q) &\rightarrow ((P \supset Q) \ \& \ (Q \supset P)) \\ (\exists x, r)P &\rightarrow (\sim((x, r)(\sim P))) \\ (\exists !x, r)P &\rightarrow ((\exists x, r)P \ \& \ (x, r)(y, r)(P \neq S_y^*P \mid \supset x = y)) \\ (A \neq B) &\rightarrow (\sim(A = B)) \\ T &\rightarrow (x, 0)x = x \\ F &\rightarrow (\sim T) \end{aligned}$$

The variable y must not be free in P .

(b) Formulas of form $A \ \& \ B \ \& \ \dots \ \& \ P$ we consider not to need any more brackets, since they have the same meaning in whatever manner the brackets are put in. Strictly speaking this equivalence only applies in virtue of rule IV below, and the reader may prefer to adopt some definite convention of his own as to the way the missing brackets are to be supplied. Similar considerations apply to formulas of form $A \vee B \vee \dots \vee P$.

(c) We shall often leave brackets out in cases where it is quite obvious how they should be replaced. Excessive bracketing often makes the formulas difficult to read. It is not thought worth while to introduce definite conventions in the

present paper: we rely on common sense instead. Likewise we permit alterations in the form of a pair of brackets. These common sense conventions have already been applied to some extent.

2. Formal account of the nested-type system. We now describe the practical system in the usual formal manner, specifying what series of symbols are to be regarded as term-formulas, proposition formulas, variables, provable formulas, etc. We do not follow this aspect very far in the present paper, believing that mathematics is suffering more from lack of sound notation than from lack of rules of procedure.

Term variables. The symbols $a, b, \dots, n, o, u, v, w, x, y, z, a', b', \dots$ are term variables.

Proposition variables. The symbols $p, q, r, s, t, p', q', \dots$ are proposition variables.

Term formulas, proposition formulas, and formulas. Term variables are term-formulas. Terms (U_1^H, U_2^H, \dots) are term formulas. Proposition variables are proposition formulas. If A and B are term formulas and P and Q are proposition formulas and x is a term-variable and r a numeral representing a non-negative integer, then (AB) and $(\iota x, r)P$ are term formulas and $(A = B)$, $(\sim P)$, $(P \supset Q)$, $D'A$, $(x, r)P$ are proposition formulas. Term formulas and proposition formulas are formulas. No expression is a term variable, term formula, proposition variable, proposition formula, or formula unless compelled to be so by the foregoing.

Free and bound occurrences of variables. Each occurrence of a variable in a formula is either a bound or a free occurrence, but cannot be both. Occurrences of proposition variables are always free. The occurrence of the term variable X in the formula X is free. In the formulas (AB) , $(\iota X, r)P$, $(A = B)$, $(\sim P)$, $(P \supset Q)$, $D'A$, $(x, r)P$ the occurrences of the various variables are free or bound according as they were free or bound in their corresponding occurrences in A , B , P , or Q except that the occurrences of X in $(X, r)P$, $(\iota X, r)P$ are bound.

It may be observed that all four possible combinations concerning the presence or absence of a variable bound or free in a formula can occur. Examples are T' , x , $(\iota x, 0)(x = x)$, $x = (\iota x, 0)(x = x)$.

Formulas and tautological formulas of the propositional calculus. The formulas of the propositional calculus are defined to be the least class of formulas containing the propositional variables, and containing $(P \supset Q)$ and $(\sim P)$ whenever it contains P and Q . Tautological formulas of the propositional calculus are those which always give the value T if a substitution of values T or F is made for the variables, and the result then evaluated as follows: $T \supset T$ is F, $T \supset F$ is F, $F \supset T$ is T, $F \supset F$ is T, $\sim T$ is F, $\sim F$ is T.

The rules of procedure (provable formulas). We word our rules of procedure in the form of a definition of the "provable formulas". Throughout, r is any numeral representing a non-negative integer.

Rule I (Change of bound variables). The formulas

$$(x, r)P \equiv (y, r) S_y^x P \mid$$

$$(\iota x, r)P \equiv (\iota y, r) S_y^x P \mid$$

are provable if P is a proposition formula in which y does not occur free, and x is not free at a place where y would be bound.

Rule II (Substitution). If P is provable, then $S_A^x P \mid$ and $S_Q^y P \mid$ are provable, where A and Q are respectively term and proposition formulas, and the bound variables of P are distinct both from x and q and from the free variables of A and of Q .

Rule III (Quantifiers). If either of the two formulas $H \supset (D'x \supset P)$, $H \supset (x, r)P$ is provable, and x is not free in H , then the other is also provable.

Rule IV (Propositional calculus). Any tautologous formula of the propositional calculus is provable.

Rule V (Modus ponens). If the formulas $P \supset Q$ and P are both provable then Q is provable.

Rule VI (Descriptions). If P is a proposition formula in which x does not occur bound, then the formulas

$$(\exists !x, r)P \supset S_{(\exists x, r)P}^x P \mid$$

$$\sim(\exists !x, r)P \supset (\iota x, r)P = C$$

$$D^r(\iota x, r)P$$

are provable.

Rule VII. The formula

$$(x, r)D^r A \supset (\exists y, r + 1)(\sim D^0 y \ \& \ (x, r) yx = A)$$

is provable provided y does not appear free in the term formula A .

Rule VIII (Axioms). For any numeral r representing a non-negative integer the following formulas numbered A1 to C2 are provable:

- A1. $C \neq T' \ \& \ C \neq F' \ \& \ T' \neq F'$
- A2. $D^0 C \ \& \ D^0 T' \ \& \ D^0 F'$
- A3. $[D^0 x \vee (D^{r+1} x \ \& \ \sim D^r y)] \supset xy = C$
- A4. $D^r x \supset D^{r+1} x$
- A5. $D^{r+1} x \supset D^r xy$
- B1. $x = x$
- B2. $(y = x \ \& \ y = z) \supset x = z$
- B3. $x = y \supset (zx = zy \ \& \ xz = yz)$
- C1. $(x, r)fx = gx \supset [f = g \vee D^0 f \vee D^0 g \vee \sim D^{r+1} f \vee \sim D^{r+1} g]$
(Axiom of extensionality.)
- C2. $(\exists i, r + 2)(f, r + 1)((\exists x, r)fx = T') \supset f(if) = T'$
(Axiom of choice.)

Rule IX (Axiom of infinity). The following formula is provable:

- C3. $(\exists h, 1)(\exists v, 0)(x, 0)(y, 0)[(hx = hy \supset x = y) \ \& \ v \neq hx]$

If we have a finite universe with N individuals instead of an infinite one we must replace rule IX by:

Rule IX_N. The following, D1 and D2, are provable:

$$D1. \quad D^0x \equiv (x = U_1^H \vee \dots \vee U_N^H)$$

$$D2. \quad U_n^H \neq U_m^H$$

where m and n are different and not greater than N .

We may make a number of remarks about these axioms and rules:

(1) Axioms D1, D2 are rather stronger than is really necessary. Instead we could use the one axiom

$$D^0x \supset (x = U_1^H \vee \dots \vee x = U_N^H)$$

which would be more nearly analogous to C3, but would admit the possibility of there being fewer than N individuals.

(2) The second formula under rule VI might have been omitted. If this had been done it would have been necessary to define a new description operator in terms of the old one in such a way that the second formula would apply for the new operator.

(3) It may be wondered why rules VI and VII do not appear under the axioms, $yx = T'$ being written for P and yx for A . If there had been any more rules of this kind they could have been replaced by axioms, by making similar substitutions, but these axioms would only be equivalent to the corresponding rule in the presence of rules VI, VII. It will now be clear why rules VI, VII cannot themselves be written as axioms.

(4) A term U_m and its corresponding formula U_m^H are not regarded as identical as they were in §1. We have introduced a distinction rather similar to the distinction between the real and complex numbers π . This distinction will be of value in any attempt to provide a formal justification of the system in terms of tables: it would then be very embarrassing to have the same notation both for a formula and its interpretation. The author has carried through such a justification in detail, together with a proof that the system is complete for the finite universe. This provides a good check that no essential axioms have been omitted. The theorem mentioned in the next section provides a similar check.

(5) Although rule III does not permit $H \supset (D'x \supset P)$ to be deduced directly from $H \supset (x, r)P$ if x is free in H , the deduction may be made indirectly.

(6) The axiom of choice is optional, i.e. we may drop this axiom and still retain a system adequate for the greater part of mathematics.

(7) We shall not carry out any proofs in this paper, but the following provable formulas are of interest:

$$x = y \supset (D'x \supset D'y)$$

$$(x, r)(P \equiv Q) \supset (\iota x, r)P = (\iota x, r)Q$$

$$(\iota x, r)A = B \supset (\lambda x, r)A = (\lambda x, r)B$$

$$(\iota x, r)D'A \supset (x, r)[((\lambda x, r)A)x = A]$$

$$D^{r+1}(\lambda x, r)A$$

$$(f, r)(g, r)[(x, r+1)(xf = xg) \supset f = g]$$

$$D^{r+1}x \equiv [(y, r+1)\{D'xy \& (D'y \vee xy = C)\} \& D^{r+2}x]$$

3. Equivalence with Church's system. The nested-type system described above may be proved equivalent, in a certain sense, to Church's simplified theory of types.³ The proof is long and tedious, and would not justify publication, but it may be of interest to give an exact statement of the equivalence theorem. The form of "equivalence" used has a certain interest in itself.

DEFINITION. A logical system 1 will be said to be *equivalent* to the logical system 2 if to each proposition-like formula A of 1 we can make correspond a proposition-like formula $A^{(1,2)}$ of 2, and conversely to each proposition-like formula P of 2 we can make correspond a proposition-like formula $P^{(2,1)}$ of 1, in such a way that

- (i) If A is provable in 1 then $A^{(1,2)}$ is provable in 2.
- (ii) If P is provable in 2 then $P^{(2,1)}$ is provable in 1.
- (iii) If A is a proposition-like formula of 1 then $(A^{(1,2)})^{(2,1)} \equiv A$ is provable in 1.
- (iv) If P is a proposition-like formula of 2 then $(P^{(2,1)})^{(1,2)} \equiv P$ is provable in 2.
- (v) If A and B are proposition-like formulas of 1 then we can prove $(A \equiv B)^{(1,2)} \equiv (A^{(1,2)} \equiv B^{(1,2)})$ in 2.
- (vi) If P and Q are proposition-like formulas of 2 then we can prove $(P \equiv Q)^{(2,1)} \equiv (P^{(2,1)} \equiv Q^{(2,1)})$ in 1.

The formula $A^{(1,2)}$ must be an effectively calculable function of A and $P^{(2,1)}$ of P .

It is understood that for each system there is defined a special kind of formulas called 'proposition-like formulas'; that every provable formula is necessarily proposition-like, and that it is a comparatively trivial matter to determine whether a formula is proposition-like or not. Specifically we may say that the statement " A is a proposition-like formula" should be equivalent to some statement of the form " $\varphi(n) = 0$ " where n is the Gödel representation of A and φ is some primitive recursive function. It is also understood that both systems "include the propositional calculus": this is required in connection with the logical equivalence signs in (iii) to (vi).

We are justified in describing this relation as the equivalence of the two systems, for the relation is transitive, symmetric, and reflexive, as I shall now show. The symmetry of the relation follows at once from the fact that interchange of systems 1 and 2 simply interchanges conditions (i) and (ii), (iii) and (iv), (v) and (vi). Reflexiveness is proved by taking $A^{(1,1)}$ to be A . Transitivity is not quite so easy. We shall have to bring in a third system 3. We will define $A^{(1,3)}$ to be $(A^{(1,2)})^{(2,3)}$ and $A^{(3,1)}$ to be $(A^{(3,2)})^{(2,1)}$. We assume conditions (i) to (vi) to hold for the pairs 1,2 and 2,3 and attempt to prove them for the pair 1,3. Because of the symmetry it is sufficient to prove (i), (iii), (v). To prove (i) we must prove $(A^{(1,2)})^{(2,3)}$ in 3 assuming A provable in 1. Now by (i) for the pair 1,2 we see that $A^{(1,2)}$ is provable in 2, and then by (i) for the pair 2,3 we get $(A^{(1,2)})^{(2,3)}$ in 3. To prove (iii) we must prove $((A^{(1,2)})^{(2,3)})^{(3,2)} \equiv A$ in 1.

³ Alonzo Church, *A formulation of the simple theory of types*, this JOURNAL, vol. 5 (1940), pp. 56-68.

Using (iii) for the pair 2,3 gives us $((A^{(1,2)})^{(2,3)})^{(3,2)} \equiv A^{(1,2)}$ (in 2), whence by (ii) for the pair 1,2 we have

$$(((A^{(1,2)})^{(2,3)})^{(3,2)} \equiv A^{(1,2)})^{(2,1)}$$

Also by (vi) for the pair 1,2 we have

$$(((A^{(1,2)})^{(2,3)})^{(3,2)} \equiv A^{(1,2)})^{(2,1)} \equiv (((((A^{(1,2)})^{(2,3)})^{(3,2)})^{(2,1)}) \equiv (A^{(1,2)})^{(2,1)})$$

and by (iii) for the pair 1,2 we have

$$(A^{(1,2)})^{(2,1)} \equiv A$$

Combining these last three results by the rules of the propositional calculus we obtain

$$(((A^{(1,2)})^{(2,3)})^{(3,2)})^{(2,1)} \equiv A$$

as required.

To prove (v) for the pair 1,3 we must prove

$$((A \equiv B)^{(1,2)})^{(2,3)} \equiv ((A^{(1,2)})^{(2,3)} \equiv (B^{(1,2)})^{(2,3)})$$

By an application of (v) to the pair 1,2 followed by an application of (i) to the pair 2,3 we get

$$((A \equiv B)^{(1,2)} \equiv (A^{(1,2)} \equiv B^{(1,2)}))^{(2,3)}$$

and by an application of (v) to the pair 2,3 we have

$$((A \equiv B)^{(1,2)} \equiv (A^{(1,2)} \equiv B^{(1,2)}))^{(2,3)} \equiv (((A \equiv B)^{(1,2)})^{(2,3)} \equiv (A^{(1,2)} \equiv B^{(1,2)})^{(2,3)})$$

Combining these by the propositional calculus gives

$$((A \equiv B)^{(1,2)})^{(2,3)} \equiv (A^{(1,2)} \equiv B^{(1,2)})^{(2,3)}$$

Condition (v) applied to 2,3 also gives

$$(A^{(1,2)} \equiv B^{(1,2)})^{(2,3)} \equiv ((A^{(1,2)})^{(2,3)} \equiv (B^{(1,2)})^{(2,3)})$$

from which we now obtain the required result.

Our definition of the equivalence of two systems could be summed up by saying that they are equivalent if we can translate from either system to the other in such a way that provable propositions translate into provable propositions again, and so that a double translation gives rise to a proposition equivalent to the original. This explanation ignores the last two conditions (v) and (vi), which are rather too tenuous for such rough handling.

The equivalence theorem then states that the nested-type system is equivalent to Church's system, if the proposition-like formulas of the nested-type system are taken to be the proposition formulas without free variables, and the proposition-like formulas of Church's system are those of type \circ without free variables.

4. Relaxation of type notation. The form of type theory which we have described is one in which the types themselves do not intrude very much. Even so they do still intrude to an appreciable extent, and it would be desirable to

see how much further they can be relegated to the background. A possible way of doing so will be described in this section.

We could sum up the effect of type theory as it appears in this system by saying that we give no meaning to the expressions 'for all x, A ', 'there exists an x , such that A ', 'the x , such that A ', 'the function whose value for argument x is A ' (usually expressed symbolically as $(x)A$, $(\exists x)A$, $(\forall x)A$, $(\lambda x)A$, respectively). Instead we give meaning to the expressions $(x, r)A$, $(\exists x, r)A$, $(\forall x, r)A$, $(\lambda x, r)A$. Nevertheless in a large class of cases we *can* assign meanings to $(x)A$, $(\exists x)A$, $(\forall x)A$, $(\lambda x)A$ in a satisfactory manner. A typical case is that of a formula of the form $(\forall x)P$ where P is such that we can prove $P \supset D^{10}x$, say. In this case for any integers $r, s \geq 10$ we can prove $(\forall x, r)P = (\forall x, s)P$ and it is therefore natural to stipulate that $(\forall x)P$ shall stand for the common value of $(\forall x, 10)P$, $(\forall x, 11)P$, \dots . We may say more generally that if $(\forall x, r_0)P = (\forall x, r)P$ is provable for all $r \geq r_0$ then $(\forall x)P$ shall be said to be interpretable and to have the interpretation $(\forall x, r_0)P$. This is of course still only the beginning of a definition of "the interpretation of a formula with some type bounds omitted." In order to give the complete definition we must deal properly with formulas having free variables: results such as $P \supset D^{10}x$ (quoted above) are not normally provable if P has free variables other than x . On this account we introduce the idea of "interpretability under hypotheses"; the hypotheses involved are usually of the form $D'x$. The complete definition is as follows:

All variables and C, T', F' provide their own interpretations under any hypotheses.

If A, B, P, Q have interpretations A', B', P', Q' under certain hypotheses, then (AB) , $(A = B)$, $D'A$, $(P \supset Q)$, $(\sim P)$ have the interpretations $(A'B')$, $(A' = B')$, $D'A'$, $(P' \supset Q')$, $(\sim P')$ respectively under the same hypotheses.

If, for each $r \geq r_0$, P has the interpretation P_r under hypothesis $H \& D'x$ where H does not contain x free and we can prove

$$(A) \quad H \supset (\forall x, r_0)P_{r_0} = (\forall x, r)P_r$$

then $(\forall x)P$ has the interpretation $(\forall x, r_0)P_{r_0}$ under hypothesis H . If instead of (A) we can prove

$$H \supset [(\forall x, r_0)P_{r_0} \equiv (\forall x, r)P_r]$$

then $(\forall x)P$ has the interpretation $(\forall x, r_0)P$ under H .

No formula has any interpretation unless compelled to by the foregoing.

It may be observed that every formula of the nested-type system is interpretable and provides its own interpretation. Also that if $H \supset K$ is provable and a formula has a certain interpretation under K then it has the same interpretation under H .

If P has the interpretation P_r under $H \& D'x$ and we wish to show either that $(\forall x)P$ has the interpretation $(\forall x, r_0)P_{r_0}$, or that $(\exists x)P$ has the interpretation $(\exists x, r_0)P_{r_0}$ under H , it is sufficient to prove $P_r \supset D'^0x (r \geq r_0)$.

It will be seen that this definition does not provide an effective means of determining whether or not an expression is interpretable. This need not be

considered a serious drawback, as we seldom need to establish that an expression is not interpretable.

The most natural cases where we can apply the above definitions are those of $(x)(A \supset B)$, $(\exists x)(A \& B)$, $(\exists x)(A \supset D^r x)$ where $A \supset D^r x$ is provable for some r_0 . It is fairly easy to remember which are the most important expressions A of this kind: e.g. in almost any formalisation we shall have “‘ x is a real number’ $\supset D^r x$ ” with $r_0 = 10$ say; this fact would be remembered in the form “the class of real numbers is all right.” It is not so easy to remember the appropriate numbers r_0 , but it is hardly necessary to do so if the notations $(x)A$ etc. are adhered to throughout. When A is such that for some r_0 we can prove $A \supset D^r x$ I shall call the class of x for which A is true a “noun-class.” There is a very close connection between the part played by the formulas A in our system and nouns in ordinary language; so much so that one might say that type theory had been instinctively obeyed for thousands of years before its discovery by Russell. This connection may be seen by translating $(x)(A \supset B)$, $(\exists x)(A \& B)$, $(\exists x)(A \supset D^r x)$ roughly as “All A satisfy B ,” “There exists an A satisfying B ” and “The A which satisfies B .” In each case A is translated in the form of a noun. It seems that the necessity to use nouns prevents us automatically from committing type fallacies in common speech. We can probably only break down this ‘safety device’ by using nouns such as ‘thing’ or ‘object’ with the intended meaning ‘anything whatever.’ In the case of the Russell paradox (‘class of all classes which are not members of themselves’) we use the word ‘class’ in very much that way. We use it to mean ‘class of anything whatever.’

There are various ways in which we might make use of the idea of interpretable formulas to transform what we have called the ‘nested type system’ into something rather more closely analogous to common mathematical practice. One possibility is simply to regard the formulas without types as abbreviations of the appropriate formulas of the nested-type system, such formulas only being used when the appropriate metamathematical result justifying the interpretation has been established. This does not seem to be really satisfactory because of the frequent need to prove such metamathematical results. Alternatively we may set up some new symbolic system in which the formulas form a considerably wider class than those of the nested-type system, and are all interpretable as defined above. The author has investigated two such systems. In one of them the expression $(x, A)P$ had the meaning which we have assigned to $(x)(Ax = T' \supset P)$. This is always interpretable if A is interpretable and without free variables. This scheme leads to rather heavy formulas in the elementary stages, though it may have advantages when more advanced branches of mathematics are reached. The second system appears rather more hopeful, and will now be described briefly. It may be called the “concealed type” system.

The formulas in the concealed type system will be described as “admissible formulas” to distinguish them from the formulas of the nested-type system. The admissible formulas will in fact be included amongst the interpretable formulas associated with the nested-type system. There will be admissible term formulas (ATF) and admissible proposition formulas (APF). We define APF, ATF, and provable formula by a simultaneous induction. Consequently there

is no rule for determining whether an expression is an admissible formula or not: this is not usual in logical systems, but there seems to be no good reason for a positive taboo on such an arrangement. We now give the inductive definitions.

Every term variable is an ATF and every proposition variable is an APF.

The symbols E, C, T', F' are ATF.

If A, B, F are ATF and P, Q, R, S are APF, and $P \supset Ax = T', \sim Q \supset Ax = T'$ are provable formulas then $(\exists x)P, (x)Q, (B = F), (R \supset S), \sim R$ are APF, and $(,x)P, (BF)$ are ATF. The variable x must not occur free in A .

Free and bound occurrences of variables are defined as in the nested-type system.

The symbol E corresponds to $(\lambda x, 0)T'$ of the nested type system. Its main purpose is to take the place of D^0 and indirectly to replace the other D' . For any formula A we can prove $((\lambda x, 0)T')A \equiv D^0A$ in the nested-type system.

If A and B are ATF not containing x, y , or z free then the two expressions below are ATF, viz.

$$(\,y)(x)[(yx \vee yx = C) \& \sim E y \& (yx \equiv (Ax \vee Bx))]$$

$$(\,y)(x)[(yx \vee yx = C) \& \sim E y \& (yx \equiv (z)\{(Bz \supset A(xz) \& (\sim Bz \supset xz = C)\})]$$

They may be abbreviated respectively to *Sum* AB and *Pot* AB . In these formulas we have adopted the useful convention that a formula of form $A = T'$ may be abbreviated to A . The context will always enable one to determine when this abbreviation has been applied. We shall continue to use this convention.

Strictly speaking the definitions of *Sum* AB and *Pot* AB are invalid because the bound variables x, y, z were not specified. This technical difficulty may be resolved by requiring x, y, z to be the three earliest variables not appearing free in A, B .

The remainder of the definition consists of the axioms and rules of procedure. It may be remembered that these took the form of a definition in the nested-type system also

Rules of procedure (concealed-type system).

Rule I. The formulas

$$(x)P \equiv (y)S_y^x P|$$

$$(\,x)P = (\,y)S_y^x P|$$

are provable if $(x)P$ is an APF in which x is not bound in P , y does not occur free, x does not occur at a place where y would be bound, and $(,x)P$ is an ATF.

Rule II. If P is provable, then $S_A^x P|$ and $S_Q^y P|$ are provable, where A and P are respectively an ATF and an APF, and the bound variables of P are distinct both from x and q and from the free variables of A and of Q .

Rule III. If $H \supset P$ and $H \supset (x)P$ are both APF and one of them is provable then the other is provable also.

Rule IV. Any tautologous formula of the propositional calculus is provable.

Rule V. If the formulas $P \supset Q$ and P are both provable then Q is provable.

Rule VI. If P is an APF in which x does not occur bound, then the formulas

$$\begin{aligned} (\exists !x)P \supset S_{(,x)P}^x P \\ \sim (\exists !x)P \supset (\cdot x)P = C \end{aligned}$$

are provable provided they are APF.

Rule VII. If A is an APF in which x, y, z, u do not occur free, then

$$(x)(ux \supset z A) \supset (\exists y)[(Pot zu)y \ \& \ (x)(ux \supset yx = A)]$$

is provable.

In rule VI the definition

$$(\exists !x)P \rightarrow (\exists x)P \ \& \ (x)(y)(P \ \& \ S_y^x P) \supset x = y$$

is understood, y standing for a variable not occurring free in P .

The axioms are:

$$A1 \quad C \neq T' \ \& \ C \neq F' \ \& \ T' \neq F'$$

$$A2 \quad EC \ \& \ ET' \ \& \ EF'$$

$$A3 \quad Ex \supset xy = C$$

$$B1 \quad x = x$$

$$B2 \quad (y = x \ \& \ y = z) \supset x = z$$

$$B3 \quad (x = y) \supset (zx = zy \ \& \ xz = yz)$$

$$C1 \quad [(Pot yu)f \ \& \ (Pot yu)g \ \& \ (x)(ux \supset fx = gx)] \supset f = g$$

$$C2 \quad (\exists i)[(Pot u(Pot Eu))i \ \& \ (f)\{(Pot Eu)f \ \& \ (\exists x)fx\} \supset f(if)\}]$$

$$C3 \quad (\exists t)[(Pot E E)t \ \& \ (\exists v)[Ev \ \& \ (x)(y)\{(Ex \ \& \ Ey)$$

$$\supset ((tx = ty \supset x = y) \ \& \ v \neq tx)\}]]$$

To complete our inductive definition we need only add that no expression is an ATF, APF, or provable formula unless compelled to be so by the foregoing.

We may say that roughly speaking type theory appears in the concealed type system only through the condition that $P \supset Ax = T'$ must be provable if $(\cdot x)P$ is to be an ATF, and a similar condition for $(x)P$. The system is related to the nested-type system by the following metamathematical results:

(1) If we substitute $(\lambda x, 0)T'$ for E throughout an admissible formula without free variables we obtain an interpretable formula.

(2) If in a provable formula of the concealed-type system without free variables we make the substitution mentioned in (1) and then form an interpretation of the resulting formula we obtain a provable formula of the nested-type system.

(3) Every provable formula of the nested-type system is obtainable as in (2).

A valuable aid in the proof of these is the following result which concerns the nested-type system only:

(4) If A is a term formula containing only the variables x_1, x_2, \dots, x_n free, and m_1, m_2, \dots, m_n are non-negative integers, then there is an integer k such that $D^{m_1}x_1 \ \& \ \dots \ \& \ D^{m_n}x_n \supset D^k A$ is provable.

Some theorems about Church's system (Three unpublished manuscripts)

There are three incomplete typed manuscripts:

- I Entitled *Consequences of the Peano Axioms*, it has pages (numbered by AMT) 37–43.
- II Entitled *Finite models of Church's and Zermelo's systems*, it has pages numbered 60–73, 84.
- III Entitled *Some Theorems about Church's system*, it consists of 19 unnumbered pages running consecutively.

PREFACE

In an undated letter (late 1941 or early 1942) to Newman, Turing writes

I have got fairly well on with 'Some syntactical theorems about Church's system'. It is rather a collection of oddments and contains

- A Proof that any typed formula has a normal form
- B Theorem about $p_o \equiv q_o \supset p_o = q_o$ and the result on the Inf Ax.
- C Result about consistency of 11^α .
- D Consistency of Axiom of Descriptions in the presence of the Axiom of Extensionality.
- E Consistency of Church's system if we omit descriptions and extensionality.
- F Connection of Church's system and Zermelo's system.

I have added the letters for later reference. In fact, A, B, D, E are dealt with, in order, in III, C in I and F in II. Turing continues the letter with a description of the method of virtual types; but this is not explicitly mentioned in III, though it is in I, and may have been set out in the missing pages.

I think it is probable that II and III were saved from an earlier draft, and that I is an incomplete final draft: possibly, he intended to continue it with a section on virtual types. First, using the letters above, I describe

Manuscript III: Some Theorems about Church's system

A Proof that any typed formula has a normal form (p. 1)

Turing may well have been the first person to write out a proof that every formula of the typed λ -calculus has a normal form. So, in view of the subsequent inter-

est in such (weak) normalisation theorems, it seems worth quoting his proof in full.

We will well-order the formulae of Church's system as follows. We say that the formula has an unreduced part of order n if it has a part of form $(\lambda x_\alpha A_\beta)B_\alpha$ where $\beta\alpha$ is of length n . If we wish to decide which of two formulae precedes we find out what is the highest order of any unreduced part in each. The formula which has an unreduced part of higher order than any part of the other comes later. Suppose however that the maximum orders are the same, n say, then the one which has the more unreduced parts of order n comes the later. But these numbers may be also equal, and in this case we compare the numbers of unreduced parts of order $n - 1$, and if we fail with this we go to the unreduced parts of order $n - 2$. If eventually there is a difference the formula with the greater number of unreduced parts comes later, if however the numbers remain the same to the end, i.e. as far as those of order 1 then the longer formula comes later. It is not difficult to see that this is a well-ordering of formulae, of type ω^ω .

Now, when we perform a reduction on a formula, in which we reduce one of the unreduced parts of highest order, we necessarily decrease the number of unreduced parts of the highest order, for we destroy one and we do not create any more; this at any rate will be the case if we choose the unreduced part of highest order whose λ lies farthest to the right. We therefore reduce the formula to one which is earlier in the sequence, and as the sequence is well-ordered the sequence of reductions must come to an end.

B The proposition $p_o \equiv q_o \supset p_o = q_o$. (pp. 2–3)

Turing gives a sketchy and not quite correct proof that this is consistent with the other axioms of Church's system; his proof makes essential use of the axiom of description at type 0. The words 'and resultant Inf $Ax.$ ' in the letter to Newman refer to the fact that if the above axiom is dropped then it is consistent to add an axiom of infinity for type 0. Turing raises this point in a letter written to Newman sometime in 1940; but it is not discussed anywhere in the manuscripts.

I remember Turing telling me that Church regarded the above proposition as a strong one, whose consistency might be hard to prove. But in a conversation in 1978, Church doubted if this had ever been his opinion.

D Descriptions (pp. 4–7)

Turing takes the axiom of descriptions for type α to be the assertion that there exists an object of type $\alpha(o\alpha)$ which acts as a description operator. Church had already shown that the axiom for type $\alpha\beta$ is a consequence of it for type α . Turing takes the proposition **B** above as an axiom; from this it follows that the

axiom of descriptions for type σ can be proved. He then replaces the type ι by the type $\sigma\iota$; after this replacement, the axiom of descriptions holds for all types. He has to check that the other axioms hold after the replacement. For the axiom of infinity **8 $^\iota$** , he uses a modification of the proof in *A Formal Theorem*; for the axiom of extensionality he points out that the axiom of descriptions implies the existence of an extensional descriptions operator.

On page 9 of the typescript, he argues in favour of taking

$$\mathbf{Inf}^2: \exists s_u \exists 0_\iota \forall x_\iota \forall y_\iota (sx = sy \supset x = y \& sx \neq 0)$$

as an axiom of infinity, rather than Church's **7** and **8**.

REMARK. What Turing showed is that proposition **B** and the axiom of descriptions (at all types) are equiconsistent – with or without the axioms **10 $^{\alpha\beta}$** of extensionality. Gandy [1956] showed that if Church's system without descriptions and extensionality is consistent then so is the system obtained by adding proposition **B** and the axioms **10 $^{\alpha\beta}$** . This result would have surprised Turing – see **E** below. (Gandy used **Inf**² rather than Church's **7** and **8**, but his argument can be modified to apply to Church's formulation.)

E The consistency proof (pp. 9–19)

Turing, following a method due to von Neumann, first reformulates Church's system by only allowing formulae which are in normal form, and only allowing substitutions to be made in the axioms. He then writes:

It will be proved in this section that Church's system is consistent if we omit the axiom of choice, descriptions and extensionality. Thus we might say that the axiom of extensionality is the 'nigger in the wood-pile', for when we have added it, we may also add the axiom of descriptions as we have shown. Unfortunately, I do not believe that Gödel's consistency proof for the axiom of choice applies to Church's system, so that we cannot take a further step and include it.

The idea of Turing's consistency 'proof' is analogous to Hilbert's ε -substitution method. With each (universal) quantifier in a proof of a formula **A** there is to be associated a finite set of terms in such a way that if each quantifier is restricted to range only over its associated set then the resulting formula is true, and hence provable directly. But Turing's recipe for constructing the associated finite sets is much too simple-minded; it would not even work for the predicate calculus.

Remarks

Sometime around 1952 Turing told me that he had written out such a proof, but did not show it to me. He also said that he had convinced Church that his proof was sound. Church, however, (in 1978) remembered that Turing had (in 1938) claimed to have such a proof, but that he (Church) had been suspicious of it and that he would only have been convinced if Turing produced a fully written-out account of the proof, which he never did.

I showed, soon after Turing's death, that the axiom of extensionality is relatively consistent, and so could not be 'the nigger in the wood-pile' that Turing claimed it to be (see Gandy (1956)). In a footnote, with natural piety, I suggested that Turing might have discovered his error. I now think this unlikely: after completing this typescript, he probably never thought hard about the matter again.

Turing's doubts about adapting Gödel's consistency proof to type theory are ill-founded; but the details would be tiresome, since one would have to use recursion along well-orderings instead of ordinals. A proof of the consistency of the axiom of choice for Peano arithmetic will be found in

Manuscript I: Consequences of the Peano axioms

C Result about consistency of 11^α

Using the axiom of infinity Inf^2 (as in **D** above) for the type of individuals, Turing defines a virtual type β (a subtype of ι) by the predicate

$$\lambda x_\iota. (f_{\iota\iota})(f0_\iota \supset ((y_\iota)(fy \supset fSy) \supset fx));$$

the objects in type β are just the natural numbers and satisfy Peano's axioms.

He proves various simple theorems, and makes connections with *A Formal Theorem*, in particular showing that Church's axioms **7** and **8** (with β replacing ι) are satisfied. He notes that the axiom of choice holds in type β , thus 11^ι is consistent.

Manuscript II: Finite models of Church's and Zermelo's systems

F Connection of Church's system and Zermelo's system

Turing writes:

We now turn to another topic connected with virtual types viz. establishing a connection between the logical system of Zermelo and Church's form of

the simplified theory of types. Zermelo's system as originally proposed by him was not described in sufficiently formal terms as judged by modern standards, and various interpretations have been proposed since (J.v. Neumann 1, Bernays 1). None of these interpretations include the abstraction idea, which is such a valuable part of Church's system, and I shall therefore propose another form of 'Zermelo system' in which this idea is embodied. (?? I shall also connect this 'Zermelo system' with the Bernays system, which was used by Gödel in his proof of the consistency of the axiom of choice and of the continuum hypothesis??).

The Zermelo system may best be introduced by considering the case of a universe containing a finite number of objects. We shall find a number of methods of argument valid for such a case. These methods of argument taken by themselves will then form the Zermelo system. This method of explanation will, I hope, make clear the meanings intended to be attached to the formulae better than could be done otherwise. Before doing this however it may be as well to give a similar discussion of the Church system in the finite case.

Turing then describes a system of tables over a finite system of individuals for Church's system, entirely similar to that given in *Practical Forms*. His version of Church's system includes axioms 7, 10 (extensionality) and 11 (choice) while axiom 8 is replaced by an axiom of finitude similar to (IX)_N of *Practical Forms*. He then proves that this system is sound and complete for the given interpretation (cf. Theorem A of *Practical Forms*).

Remarks

The references must be to von Neumann (1925) and Bernays (1937-). I do not think that Turing had read Skolem (1922). Turing's 'Zermelo system' is the nested-type theory of *Practical Forms*. I do not think that Turing ever realised that, in the presence of axioms of infinity, Zermelo's set theory is noticeably stronger than type theory: in the former one can give a truth definition for the latter. I also doubt that he realised that the hereditarily finite sets over a finite set of *urelemente* provide a model of Zermelo–Fraenkel set theory without the axioms of infinity and foundation.

It is typical of Turing's down-to-earth style of thought that he uses finite systems as a key to the understanding of infinitary systems. In conversation he would stress this as being the right line of approach.

This Page Intentionally Left Blank

A Practical Form of Type Theory II

(Unpublished manuscript of 81 unnumbered pages of typescript together with 22 manuscript pages inserted in various places.)

Description

The first sentence is:

In this paper theorems A and B enunciated in Part I are proved.

Theorem A (of *A Practical Form of type theory*), or, as in remark (4) of S2 of *Practical Forms*, states that the nested-type system, with an axiom of finitude (Rule IX_N of *Practical Forms*) is sound and complete for the model whose elements in any type *r* are the tables of type *r* over a set of individuals of cardinality *N*.

Theorem B is the equivalence, as defined in S3 of *Practical Forms*, of Church's system (system 1) with the 'practical system' (system 2) – that is the nested-type system as formulated in *A Practical Form of type theory*.

There are seven numbered sections; I give their titles together with some remarks about them.

1 Preliminary definitions and conventions (pp. 1–17)

Church's system and the practical system are summarised. A large number of conventions, for abbreviating formulae and for setting out proofs are given. *A propos* of the conventions for using dots (much as in *Dots*), Turing writes:

These conventions may seem rather alarming, but it is recommended that not too much notice should be taken of them, and that the bracketing be treated on a common sense and contextual basis.

The section ends with statements of Theorem A and B

2 Formal development of the [practical] system (pp. 18–23A)

3 *Justification and completeness in the finite case (Theorem A)* (pp. 24–33)
For any table *U*, based on *N* individuals, Turing gives the corresponding term formula *U^H* (cf. Remark (5) in S2 of *Practical Forms*). The nub of both the soundness and completeness proofs is, in effect, the demonstration of a (metamathematical) lemma:

For any tables *U*, *V*, *W* over *N* individuals *UV* = *W* iff $\&-\mathbf{N} U^H V^H = W^H$, where $\&-\mathbf{N}$ means 'is a theorem of the nested-type system with Rule IX_N'.

The significance of this theorem is discussed by Turing (and by me) in *Some Theorems*.

4 Church's system derived from the practical system (pp. 34–49)

Turing defines a translation from Church's system into the practical system and shows that it preserves derivability and equivalence (i.e. clauses (i) and (v) of S3 of *Practical Forms*). In the last two pages there is a description of the method of virtual types; for quotations from this see S3 of the *General Introduction*.

5 The practical system developed in the Church system (pp. 50–62)

Naturally, the going here is heavier than in S4. Turing constructs a map from the cumulative type r into a (sufficiently high) single Church type r' ; in fact the choice of r' may depend on the context in which D^r occurs. There is a long list of definitions, and mostly unproved lemmas which follow from them. Thereafter the proofs of derivability and equivalence ((ii) and (vi) of S3) are straightforward, but long.

6 Continuation of Theorem B (pp. 63–70)

This gives a proof of condition (iii) of S3.

7 Conclusion of Theorem B (pp. 71–81)

This gives a proof of condition (iv) of S3.

Comments

From *A Practical Form of type theory* we know that Turing originally intended to publish this paper; work on it will have been concurrent with that on *A Practical Form of type theory*. Both papers must have been more or less finished during, or before, 1944 (cf. the Introduction to *Reform*). From the various connections, and remarks such as ‘omit this?’ it is plain that Turing regarded it as a first draft and then decided not to publish it for obvious reasons: the labour of preparing a final draft would have been considerable, the correctness of the results was not in question, and the methods used were fairly straightforward. In *Practical Forms* (p. 89) he writes, of Theorem B, ‘The proof is long and tedious and would not justify publication . . .’.

A quite separate item is a manuscript of 31 pages with the heading

Outline proof of metatheorem p. 35 of ‘Practical forms’

The metatheorem(s) in question are (1), (2) and (3) of the last paragraph of *Practical Forms*. The manuscript was evidently written at speed. After a short canceled paragraph on p. 1A he has written ‘3hrs 16.4 sec’. I do not know to what

this refers – it could be the whole MS. The use of a stop watch (as on a training run) is plainly something of a joke. The manuscript must have been written (in 1946?) when Turing was working on a final draft of *Practical Forms*.

This Page Intentionally Left Blank

The Reform of Mathematical Notation and Phraseology

(unpublished manuscript ca. 1944)

PREFACE

This is an unfinished typescript with pages numbered 1 to 18, but pages 9 to 12 are missing; they must have contained an account of the nested-type system (and perhaps of the concealed-type system) of *Practical Forms*. I remember that Turing was writing this paper when we were together at Hanslope Park, from Spring 1944 to Spring 1945. Evidently Turing had previously worked out the nested-type and concealed-type systems; presumably he had already completed the first draft ('A Practical Form of type theory') of *Practical Forms* and a draft of *A Practical Form of type theory* II.

Some of the logical points which Turing discusses are now familiar to 'the mathematician-in-the-street'; and there is a certain amount of repetition. But, after some hesitation, I have decided to reprint the whole – excluding of course those parts that Turing himself deleted.

Comments

1. The Deduction Theorem

In the 30's and 40's this theorem was not as familiar to the mathematician with an interest in logic as it is today. In *Principia Mathematica*, theoremhood not derivability is the primitive notion, so that 'B can be logically derived from A' is interpreted as $\vdash A \supset B$; hence the steps in a proof, which would most naturally be regarded as derivations from hypotheses, take the form $A_1 \supset A_2 \supset \dots \supset B$ where the A_i (usually written in highly abbreviated form) are the hypotheses in force. (The same is true of Quine's 'Mathematical Logic' which Turing was reading in 1944.) The theorem is not mentioned in the first edition of Hilbert and Ackermann (1928). Turing will have learnt of it from Church's lectures. When I failed to understand it – see the Introduction to *Practical Forms* – he referred me to the first volume of Hilbert and Bernays [1934–1939]. He regarded it as fundamental to the understanding of the logic of mathematical arguments and took it as a primitive rule in *A Practical Form of type theory*'. And, as he explains, he considered the accompanying notion of 'variable restricted by hypotheses' as equally important; even universal constants such as 'e' and 'π' can be regarded as variables which are permanently restricted by their definitions, regarded as

hypotheses. Turing was not familiar with Gentzen's (1934) system of natural deduction.

2. Functional Abstraction

I am slightly surprised that Turing does not mention Church's λ notation in this paper. He certainly used it frequently in conversations with me *circa* 1950. It is to be deplored that, even today, it has not (unlike \forall and \exists) become a standard part of mathematical notation.

3. Noun-Classes

(1) In 1929 Wittgenstein said:

When Frege and Russell spoke of objects they always had in mind things that are, in language, represented by nouns, that is, say, bodies like chairs and tables

(Wittgenstein [1979], p. 41).

He also argues that the proposition $(\exists x).\phi x$, with an unrestricted range for the variable must be nonsense (ibid, pp. 38–40 and p. 44.)

In his 1939 lectures on the Foundations of Mathematics, which Turing attended, the matter is discussed twice:

This business of “the property of being a man on this sofa” is a terrible muddle – “This is a horse”. But “Here is an x which is a horse” – no.

(Wittgenstein [1976], p. 167.)

The truth is that the way of writing a generality

$(\exists x).\phi x$

is taken from ordinary language. Only in ordinary language we never say, “There is a thing which is a man and has grey trousers.” We never talk about bare individuals. We say instead, “There is a man who has grey trousers.” (ibid., p. 268.)

(2) In the first part Turing discusses his concept of a noun class in a general way, without attempting to give a precise definition. In the second part the discussion is in terms of the nested-type system. Here A is, by definition, a noun-class if, for some r , $A \subseteq \{x = D^r x\}$ is provable. This means, as Turing points out, that the term ‘group’ does not define a noun-class. Of course, particular kinds of group do; for example, since simple Lie groups have the power of the continuum, they will form a noun class. Turing himself was always more interested in relatively concrete notions than in completely abstract ones. And I am sure that he thought of the type 0 as having at most countably many individuals. However,

there is no axiom of limitation of size, so that in fact general statements about abstract groups can be proved in the nested-type system with ‘group’ being understood as referring to a subset of type 0 together with a binary function on it. The modern view is, I suppose, that the elements of a group must form a collection, which in some standard or extended system of set theory can be proved to be a set. I think that (in 1944) Turing’s view would have been that such things (e.g., Jonsson groups) were no part of the stock-in-trade of the mathematician-in-the-street.

4. Conclusion

Perhaps the most striking thing about this paper is its modesty. 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 perform 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 in Part I) Turing disliked those high priests of logic who sought (like Quine in his ‘Mathematical Logic’) to blind the mathematician-in-the-street with arcane formalisms.

This Page Intentionally Left Blank

The Reform of Mathematical Notation and Phraseology

A.M. Turing

It has long been recognised that mathematics and logic are virtually the same and that they may be expected to merge imperceptibly into one another. Actually this merging process has not gone at all far, and mathematics has profited very little from researches in symbolic logic. The chief reasons for this 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, which are at present exceedingly unsystematic, and constitute 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. 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 '.

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.

Although it is not desirable to try and put mathematics into the straight-jacket of a logical system, it may be desirable to use such a system when investigating notation. One is likely to be taking typical phrases from mathematical text-books and analysing their meaning. It is useful to have a logical system for expressing these meanings in a fairly unambiguous way. It may not greatly matter what system is used for this purpose, and it would be quite possible for different workers to use different systems.

To be specific I am inclined to suggest the following programme

- i) An extensive examination of current mathematical and physical and engineering books and papers with a view to listing all commonly used forms of notation.
- ii) Examination of these notations to discover what they really mean. This will usually involve statements of various implicit understandings as between

The Reform of Mathematical Notation and Phraseology

- writer and reader, it may also include the equivalent of the notation in question in a standard notation.
- iii) Laying down a code of minimum requirements for desirable notations. These requirements should be exceedingly mild. In my opinion the points which should be covered by this code should include the following
 - a) Free and bound variables should be understood by all and properly respected.
 - b) Some sort of provision should be made for falling in line with the theory of types. This assumes a Russelian Weltenscheung, as applies I think to the majority of mathematicians-in-the-street.
 - c) The deduction theorem should be taken account of, i.e. it should be recognised that numerous forms of argument consist in one form or another of applications of the deduction theorem. The deduction theorem should therefore be as well known as the rule for integration by parts.
 - d) Very clear statements of the fundamental nature of the symbols should be made. There should be no danger of mistaking a real variable for a function taking real values.
 - iv) New notations suggested by symbolic logic.
 - v) Examples of the development of comparatively elementary parts of mathematics in obedience to the new code and embodying the new notations. These examples should only incorporate the new notations in cases where great advantage results. The effects of the various independent reforms should be shown separately, so far as possible, to facilitate their independent adoption.

Free and bound variables. Deduction Theorem. Constants and parameters

In this section a) and c) of iii) above will be examined in greater detail.

The symbols used in mathematics may be classified as follows.

- i) Symbols used entirely in punctuation, such as (,), 'etc.

These do not concern us at the moment.

- ii) Absolute constants, typified by Σ , 1, $=$, $+$, \log etc, etc.

These also do not concern us at the moment.

- iii) Letters representing constants, usually taken from the beginning of the alphabet, e.g., 'Let a be the radius of the sphere'

- iv) Letters representing genuine variables, for which substitution may be made, e.g., x in $x = x$.

- v) Letters for which we can substitute other letters (most of them) to get a true result, but certainly cannot substitute constants; in the latter case nonsense results. Examples are provided by the occurrence of x in

$$\int_0^1 x \, dx = \frac{1}{2}$$

and in ‘for all numbers x greater than 2, $x^2 > 3$ ’. Substitution of 1 for either of these yields nonsense.

Letters described under iv) and v) above are known respectively as free and bound variables. Free variables are really comparatively rare. This is because we do not often make statements such as ‘ $x = x$ ’ but more often something like ‘for all real numbers x , $x = x$ ’: in this the opening phrase ‘binds’ the variable. Thus x is bound in the whole statement, but is free in the part $x = x$. The difference between the constants iii) and the free variables is somewhat subtle. The constants appear in the formula superficially as if they were free variables, but we cannot substitute for them. In these cases there has always been some assumption made about the variable (or constant) previously; thus we may have the equation

$$v = \frac{4}{3}\pi a^3$$

in which we cannot substitute for v and a , these being constants because we have made these assumptions about them ‘ a is the radius and v is the volume of the sphere’. The ‘deduction theorem’ states that in such a case, where we have obtained a result by means of some assumptions, we can state the result in a form where the assumptions are included in the result, e.g., ‘If a is the radius and v is the volume of the sphere then $v = \frac{4}{3}\pi a^3$ ’. In this statement a and v are no longer constants. We are now able to substitute for them: we might substitute v for a and get a statement with the same meaning, or we could substitute 2 for both a and v getting a true statement, but one of rather unorthodox character. This process whereby we pass from P proved under an assumption H to ‘If H then P ’ may be called ‘absorption of hypotheses’. The process converts constants or ‘restricted variables’ into free variables. Variables whose character changes in this way from restricted to free usually seem to be described as ‘parameters’, although it is very difficult to give any very definite meaning to the term.

Theory of types. Domains of definition

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 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.

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'.

The most important places where this matters are

- 1) In connection with the word 'all'. We may for instance say 'All real numbers x have the property...' but not 'All things...'. In particular we should avoid putting the former in the form 'For all things x , if x is a real number...'.
2) In connection with 'there exists'. We allow 'there exists a real number such that...' but not 'There exists a thing x , such that x is a real number and...'.
3) In connection with descriptions. We allow 'The real number x such that...' but not 'The thing, such that x is a real number and...'.
4) In connection with abstraction, but this may be considered a special case of 3).

All this leaves open the question as to what are to be regarded as appropriate nouns to take the place of 'real number' in the above examples. This would probably be really treated on a fairly common sense basis, but the following rules certainly apply. The word *noun-class* is used to mean a class such as the class of real numbers in the examples above.

The sum of two noun-classes is a noun-class.

A sub-class of a noun-class is a noun-class.

The class of functions with arguments in one noun-class and values in another is a noun-class.

These rules do not lead to any noun-classes unless we know of some already. To many logicians it will be sufficient to add 'the null-class is a noun-class', but such a procedure will not be satisfactory to the average mathematician, who has but little concern with the null class, and certainly does not propose to build up the integers from it. The sensible thing to do seems to be to take for granted certain noun-classes such as the integers, and possibly also the real numbers and the points of three-dimensional space. In fact we may take as given noun-classes any classes of objects which, in the branch of mathematics concerned are usually

considered as given *a priori*. Such assumptions, combined with the above three rules will be found quite adequate.

Although it is not intended that symbolic logic should take the place of English connecting matter in proofs it is as well to be able to express anything symbolically if required. The usual form of expression of 'For all x, \dots ' is ' $(x) \dots$ '. As we have seen this is fundamentally unsound. Instead we want a notation for 'For all integers x, \dots '. For this I propose $(x, \text{Int}) \dots$, where Int is the notation used for the class of integers. Likewise we may use $(\exists x, \text{Int})$ for 'There exists an integer x ' and (x, Int) for 'The integer x '. It may also be desirable to have a notation for 'The class of functions whose arguments are integers and values real numbers'. No notation for this is suggested at the present moment.

[the next 4 pages of the typescript are missing]

3. *Discussion of the system. Application to normal mathematics*

Let us now try to picture what would happen if we try to develop such subjects as the theory of real numbers or set theory in this system. How will it differ from the normal 'straightforward rigorous' development? Apart from the fact that implications etc. are symbolically expressed the only real difference will be that instead of saying 'for all x, \dots ' we shall have to say 'for all x in segment 15...' (say) which may be taken as the way we read $(x, 15) \dots$. Now in most cases the statements to be transformed will be something more like 'For all real numbers x, \dots ' or 'For all x , if x is a real number, then...'; in such a case it will not really be necessary to express the condition that x belongs to any particular segment, for the real numbers will have already been defined so as to be all included in some one segment. In all such cases therefore we agree to omit this phrase. Likewise in descriptions instead of saying 'The unique member of segment 15, which is a real number and...' we may say 'The real number such that...' on the ground that all real numbers are members of segment 15. It may perhaps be objected that this omission will sometimes give wrong results because the meaning of the formula may depend on the segment quoted. This is so, but it will normally be the case that an increase in the values quoted will not affect the meaning of the formula; by this I mean that the formula may be proved equivalent to any formula which can be obtained from it by increasing the segment bounds. A formula without free variables, which can be proved equal or equivalent to any formula obtained from it by increasing the segment bounds, may be called a regular formula. It should be noticed that even if we had not explicitly excluded them there would not usually be any reason for expecting formulae with free variables to be regular. The explicit exclusion simplifies matters. We should try to avoid the use of formulae, which are not regular. This can always be done

essentially by the following device if necessary: suppose that somewhere there occurs (x, s) and that increasing s alters the meaning; then we may write instead $(x, s)D^s x \supset \dots$.

It may perhaps be argued that $(x, r) \dots$ means just the same as $(x)D^r x \supset \dots$, both being rendered into English by the phrase 'For all x in segment $r \dots$ ' or 'All members of segment r have the property that \dots '. This is true, and the only reason for our using this different notation is that we thereby automatically ensure that some condition $D^r x$ will be included in the proposition. It is perhaps worth making a comparison here with the syntax of ordinary language, where we have two distinct types of property, adjectives and nouns; the mathematician is inclined to regard this distinction as unreal and arbitrary, and so in a sense it is, but it does have the effect that it is impossible to refer to anything without associating a 'noun property' with it, and so if we compare the nouns with the segments we see that ordinary language has a remarkable tendency to respect the theory of types. We can make use of this fact to help make mathematics sound from the point of type theory. If each of the nouns used in mathematics defines a set completely contained in some segment, then ordinary syntax will keep us straight. I shall not of course attempt to give a formal proof of this, the informal character of language making it inappropriate: assuming however that this principle is essentially correct let us go further and see how much modification in common practice will have to be made; it actually amounts to surprisingly little. We have to be sure that the nouns used are 'legitimate nouns'. It is probably best therefore to manage with comparatively few nouns. We can make up some simple rules for the construction of legitimate nouns.

These rules may be taken to be

- i) The set of functions of one noun that have values in another noun is a noun.
- ii) The things, which are either one noun or another noun, may be collectively described by a third.
- iii) We may use a noun to describe a subset of the things described by another noun.

To this we might perhaps add that the word 'individual' may be used as a noun.

It may now be seen our nouns are essentially the 'mengen' of Zermelo and the 'sets' of Bernays and of Gödel. However in this discussion I am not trying to set up a formal system, but merely to suggest how normal rigorous mathematics can be made to take account of type theory without serious upheaval.

A glance through a number of mathematical books provides the following examples of nouns.

- a) Number, real number, integer, complex number, point, line, plane, manifold, operator, curve.
- b) Group, ring, algebra, base, polynomial.
- c) Set, class, pair, object, element.

- d) Frontier, conjugate, derivative.
- e) Integral, expression, equation, series, sequence, term.

The nouns listed under a) may be regarded as the most reasonable kind. Those under b) are like those under a) but are more complex in meaning, and might in some contexts be used illegitimately. Let us take the word group for example, and for the sake of argument let us pretend that it means what is normally meant by 'group of finite order'. If then we define a group as follows 'A group is a pair consisting of a finite class G of integers and a function $K(x, y)$ defined for x and y in G , and such that for all x, y, z in G , $K(x, K(y, z)) = K(K(x, y), z)$ and if $K(x, y) = K(x, z)$ then $y = z'$ the word group will be legitimate. If however the words 'of integers' are omitted such a wide range of possibilities is admitted that the expression is no longer legitimate. Of course the inclusion of these words does not vastly hamper group theory, but they are usually omitted for the sake of the extra generality apparently obtained. It would of course also be legitimate to use instead of the phrase 'of integers' 'of integers or classes of integers' or restrict G by some other 'legitimate noun', but to do so would not give the same intellectual satisfaction as leaving every possibility still open; if however we take this last course the word group will not be a 'legitimate noun'. The examples given under c) are not normally legitimate unless used in such phrases as 'set of points', 'pair of real numbers' etc. The word 'object' is of course the most serious offender, being in fact the outcome of an attempt to evade the salutary restrictions of English syntax. The examples d) really represent functions, because they are used in the form 'the frontier of ...' etc. However these functions take values restricted to one segment, and so may be regarded as legitimate. The examples e) may be regarded as purely syntactical, i.e. they do not themselves describe mathematical entities, but rather mathematical expressions, and therefore do not appear in the mathematical argument proper, and may be ignored for our purposes.

The difficulty in the above example concerning 'group of finite order' may be resolved to some extent by using the phrase 'member of segment N ' in place of 'integer'. It is then understood that one is at liberty to substitute any object for N throughout the book (say).

This Page Intentionally Left Blank

Part III

The Enigma, Mysteries and Loose Ends

This section was suggested by Andrew Hodges as an opportunity to take some new directions and tidy up some loose ends and it begins with his foreword to the elusive *Enigma* paper first written in 1940 but not released until 1996.

It is followed by some further comments by Jack Good on Minimal Cost Sequential Analysis and Turing's invention of this for use in *Banburismus*, a crucial statistical technique involved in breaking the Enigma code.

Martin Campbell-Kelly, an expert on the history of Computer Science, writes about Turing's Programmers Handbook, written for the first digital computer, set up at Manchester University in 1948.

Part III finishes with one of the insoluble mysteries – what might Turing have contributed to Physics if he had been able to follow up the ideas he was known to have been developing just before he died. These ideas were brought up by Robin Gandy in the letter to Max Newman which appropriately closes this part of the Volume. The letter is preceded by Andrew Hodges' stimulating discussion which connects with but continues beyond that in [Hodges 1983], pp. 512–514.

This Page Intentionally Left Blank

Turing's Treatise on the Enigma

PREFACE (by Andrew Hodges)

In April 1996 a mass of hitherto secret material was released by the National Security Agency of the United States Government. Amongst them, with reference number NR 964, Box 201, RG 457 was the item described as 'Turing's Treatise on the Enigma.' This became available from the National Archives and Records Administration, Washington DC.

The document as released is a photographic copy of some 140 pages of typescript (some pages being missing or mis-numbered). It is not signed or dated but is unquestionably in Turing's own handwriting and (poor) typewriting, with drawings also in his own hand. It is safe to identify it with what was known as 'Prof's Book' at Bletchley Park both according to Joan Murray [12], and also according to a 1945 report, *The History of Hut Eight*, by A.P. Mahon [10], which was released at the same time.

The unhelpful NSA label is 'ca. 1939–1942,' but the date of composition is probably autumn 1940. Joan Murray's article describes it as composed after she joined in June 1940. Mahon's report dates it as written in 1940. (Probably it was not transmitted to the United States until 1943, since another newly released report [1] indicates that the U.S. Army analysts did not know of Turing's early role in Bombe design until that year. Possibly however U.S. Navy analysts knew of it through their work at Bletchley Park in 1942.)

The background

Variants of the Enigma cipher machine had been used by the German armed services since the 1920s. From September 1938 Turing had given part-time assistance to the British cryptanalytic organisation, the Government Code and Cypher School. Their work was transformed by the transfer of information from Polish mathematical cryptanalysts in July 1939. From 4 September 1939, Turing worked full-time at the GC&CS war-time headquarters, Bletchley Park.

During 1939–40 Turing was foremost in developing logical, statistical and mechanical methods which allowed rapid decryption of some Enigma cipher traffic in 1940. The vital naval ciphers resisted decryption, mainly through an extra complexity (a bigram substitution) in the key-system. Turing took charge of a section (Hut Eight) devoted to this problem. Successful day-to-day decryption (of the main naval cipher) was only achieved in mid-1941. Turing's methods came to be

incorporated in efficient British and American organisations which in the later stages of the war were able to monitor much of the German communications system with momentous consequences [6,7]. But this text only describes the analysis of the Enigma ciphers in 1939–40.

Content of the document

The text appears to have been written as a guide for those starting Enigma work. But as so often, Turing’s style is hard to categorise. It is written as narrative history as well as a practical handbook; yet as history it is unsystematic, only sometimes stating the source of ideas. One must recall the total secrecy of the work, and that Turing was typing during the battle of Britain when questions of past credits and future historians must have seemed secondary. The text has the appearance of having been composed while Turing typed it, and never revised.

Pages 1–95 describe hand methods of analysis, including the solving of the wheel wirings. Turing’s text does not relate the methods described to methods used by the Polish analysts [9]; it would appear that Turing is here describing methods that he himself (or in collaboration with the other British analysts) worked out in the period before the Polish revelations, but this is not specifically asserted.

Pages 96–123 describe the machine methods which triumphed, principally the Bombe as designed by himself, G.W. Welchman and the British Tabulating Machinery engineer H. Keen in 1939–40. The Bombe was central to the entire Enigma-breaking work conducted at Bletchley Park and (later in the war) in the United States. Again no comment is made relating it to Polish mechanical methods. Turing does however describe in detail the origin of the two original logical principles which gave the Bombe its power:

- (1) the idea of using a ‘diagonal board’ to exploit the self-inverse nature of the plugboard complication, and
- (2) ‘simultaneous scanning’: the following of all possible false implications of a false hypothesis about plugboard settings.

The ideas are independent, as is brought out in Good’s discussion [5]. However in another account Good [3] presented Turing’s idea for (2) as following ‘shortly after’ Welchman’s idea for (1), and my own 1983 account [8] followed this order of presentation. Welchman’s account of (1) in *The Hut Six Story* [13] is similar. Joan Murray’s first-hand account [12] is more ambiguous as to the order of ideas, and so it is noteworthy that Turing’s own explanation definitely introduces (2) as preceding (1).

Pages 129–142 describe Turing’s early attack on naval Enigma ciphers, which Turing continued to head until late 1942. The problem revolved around the sys-

tem of bigram substitutions, as described by Good [3,5]. In this section (and only in this section) Turing clearly states the legacy of the Polish analysts.

This final section is however curiously short and inconclusive. At this point I refer to *The History of Hut Eight* by Mahon [10], which covers the entire period until 1945, and adds a good deal on Turing's work in 1939–1940 for which the principal source must have been Turing himself. There are at least three topics in this Mahon report, conspicuous through absence or vagueness in Turing's own text.

(1) Like Good [3,5], Mahon stresses the centrality of Turing's *Banburismus* method in the success of Hut Eight after spring 1941. This could produce a partial identification of the wheel order for the day's traffic, thus greatly reducing the time spent on Bombe search. Mahon gives a detailed example of its application, and states (on page 14) that Turing's report gives the theory of it. However Turing's text has only a parenthetic description of *Banburismus* (on page 134). Possibly, therefore, there is a continuation of Turing's account of naval Enigma methods, yet to be revealed in another version or copy of the report.

(2) Mahon also stresses the frustration of the 1940 period when the *Banburismus* method could not be brought into use for lack of sufficient captured material, and that the analysts pressed the navy to undertake captures. There is nothing of this in Turing's text.

(3) Mahon also reports on an internal argument persisting throughout 1940 between those guessing probable words in plaintext (in Hut Four), and Turing's group in Hut Eight. According to Mahon, Turing was a 'lamentable explainer' of their side of the argument. Turing's own report does not refer to any such difficulties.

Mahon also illustrates the context within which Turing operated, in particular the 1939–40 atmosphere of 'defeatism' regarding the prospects of Enigma decipherment. Another memoir writer [11] recalls that 'as late as the summer of 1940 I myself heard Commander Denniston, head of GC&CS, saying to the Head of Naval Section, "You know, the Germans don't mean you to read their stuff, and I don't suppose you ever will." As regards the naval Enigma, Mahon quotes Turing as saying 'no-one else was doing anything about it and I could have it to myself.' Mahon says, of a moment 'at the end of 1939':

Turing had in fact solved the essential part of the indicator system and that same night he conceived the idea of *Banburismus* "though I was not sure it would work in practice, and was not in fact sure until some days had actually broken."

Turing's own report does describe his guessing of the basic bigram system (page 136), but without a mention of *Banburismus*, and with a lack of emphasis on the breakthrough that it represented. It is hard to remember on reading

his text, that Turing's taking on and persisting with the Enigma variant of such significance in the battle of the Atlantic, and doing so alone against the tide of prevailing wisdom, was perhaps his greatest individual contribution to history. In this respect he was certainly 'a lamentable explainer.'

As regards Turing's mathematical work, the text as we have it presents none. The group-theoretic algebra is described for non-mathematicians; only elementary probability considerations are brought into play and detailed calculations are not given. The text does not mention the log-odds assessment of weight of evidence, as further developed in 1941 by Turing and Good [3–5].

The comments above should indicate the many difficulties of evaluating and interpreting Turing's text. Even Mahon, on the spot in 1945, reported finding it hard to collate an accurate history when few records had been kept. But the release of further related documents, and the detailed reconstruction of methods by a number of researchers, (including the actual rebuilding of Bombe machinery at the Bletchley Park Museum), should make it possible one day to write a definitive assessment of Turing's role in the Second World War.

Acknowledgments

I am indebted to Tony Sale, curator of the Bletchley Park Museum, for his assessment of the document and his assistance in the preparation of this description of it. In particular he has made available to me his 1998 work (as yet unpublished) which reconstructs detail lost in the copying process. I am grateful also for comments by Ralph Erskine, Philip C. Marks and Frode Weierud who at the time of writing (1999) are completing their own annotated edition of Turing's report.

I am also indebted to Lee Gladwin of the National Archives and Records Administration, for finding and copying the released documents. He has published his own survey of the context of the document [2].

References

- [1] William F. Friedman, Report on E Operations of the GC & CS at Bletchley Park, 12 August 1943; released by the National Security Agency, Washington DC, 1997; reference NR 3620, Box 1126, National Archives and Records Administration.
- [2] Lee A. Gladwin, Alan Turing, Enigma, and the Breaking of German Machine Ciphers in World War II, Quarterly of the National Archives and Records Administration, Washington DC, 29, 202–217 (1997).
- [3] I.J. Good, Enigma and Fish, in [7].
- [4] I.J. Good, Studies in the History of Probability and Statistics. XXXVII. A.M. Turing's statistical work in World War II, Biometrika 66, 2, pp. 393–6 (1979), reprinted also in the Collected Works (see [5]).
- [5] I.J. Good, Introductory Remarks for the article in Biometrika 66 (1979) in the volume Pure Mathematics (ed. J.R. Britton) of the Collected Works of A.M. Turing (North-Holland, 1992).

- [6] F.H. Hinsley et al., *British Intelligence in the Second World War*, HMSO (1979, 1981, 1984, 1988). Appendix 30 to the final volume adds much to the treatment of the 1939–40 period in the first.
- [7] F.H. Hinsley and Alan Stripp (eds.), *Codebreakers* (Oxford University Press, 1993).
- [8] Andrew Hodges, *Alan Turing: the Enigma* (Bennett, London and Simon & Schuster, New York, 1983; new edition Vintage, London 1992).
- [9] W. Kozaczuk (tr. C. Kasperek) *Enigma... (Arms and Armour Press, 1984).*
- [10] A.P. Mahon, *The History of Hut Eight, a report of 1945* released by the National Security Agency, Washington DC, 1996; reference NR 4685, box 1424, RG 457 National Archives and Records Administration.
- [11] Christopher Morris, *Navy Ultra's Poor Relations*, in [7].
- [12] Joan Murray, *Hut 8 and naval Enigma, Part I*, in [7].
- [13] G.W. Welchman, *The Hut Six Story* (Allen Lane, London; McGraw Hill, New York, 1982).

Excerpts from the “Enigma Paper”

Excerpt 1

Page 97 contains the heading ‘A mechanical method. The Bombe’ and reveals the crucial idea that defeated the plugboard complication.

Turing first gives a sample 25-letter Enigma cipher text with its ‘crib’ or guessed plaintext set against it.

The ciphertext is D A E D A Q O Z S I Q M M K B I L G M F W H A I V
The plaintext is KE I N E Z U S A E T Z E Z U M V O R B E R I Q T
(keine Zusätze zum Vorbericht)

Turing explains: ‘...a method of solution will depend upon taking hypotheses about parts of the keys and drawing what conclusions one can, hoping to get either a confirmation or a contradiction...’ The method depends absolutely upon making a correct guess about the corresponding short piece of plaintext.

DECLASSIFIED

by HAROLD DUNN

is given by the column of the inverse rod set-up, and we can find all possible positions where the click-coupling occurs from the Turing sheets or the Jeffreys sheets. In some cases there will be other constatations which are made up from letters supposed to be unsteckered because they occur in the click, and these will further reduce the number of places to be tested.

These methods have both of them given successful results, but they are not practicable for cases where there are many Steckers, or even where there are a few Steckers and many wheel-orders.

A mechanical method. The Bombe.

Now let us turn to the case where there is a large number of Steckers, so many that any attempt to make use of the ~~few~~ sixteenth unsteckered letters is not likely to succeed. To fix our ideas let us take a particular crib.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
D	A	E	D	A	Q	0	Z	S	I	Q	M	M	K	B	I	L	G	M	P	W	M	A
K	E	I	N	E	Z	U	S	A	K	T	Z	K	Z	U	M	V	O	R	B	M	R	I
24	25																					
I	W																					
Q	T																					

Presumably the method of solution will depend on taking hypotheses about parts of the keys and drawing what conclusions one can, hoping to get either a confirmation or a contradiction. The parts of the keys involved are the ~~maximally~~ wheel order, the rod start of the crib, whether there are any turnovers in the crib and if so where, and the Stecker. As regards the wheel order one is almost bound to consider all of these separately. If the crib were of very great length one might make no assumption about what wheels were in the L,H,W, position and M,W, position, and apply the method we have called a 'Stecker knockout' (an attempt of this kind was made with the 'Feindseeligkeiten' crib in Nov. '39), ~~maximally~~ or one might sometimes make assumptions about the L,H,W, and M,W, but none, until a late stage about the R,H,W. In this case we have to work entirely with constatations where the H,W,W, has the same position. This method was used for the crib from the ~~large~~ Schlüsselzettel of the Vorpostenboot, with success; however I shall assume that all

Excerpt 2

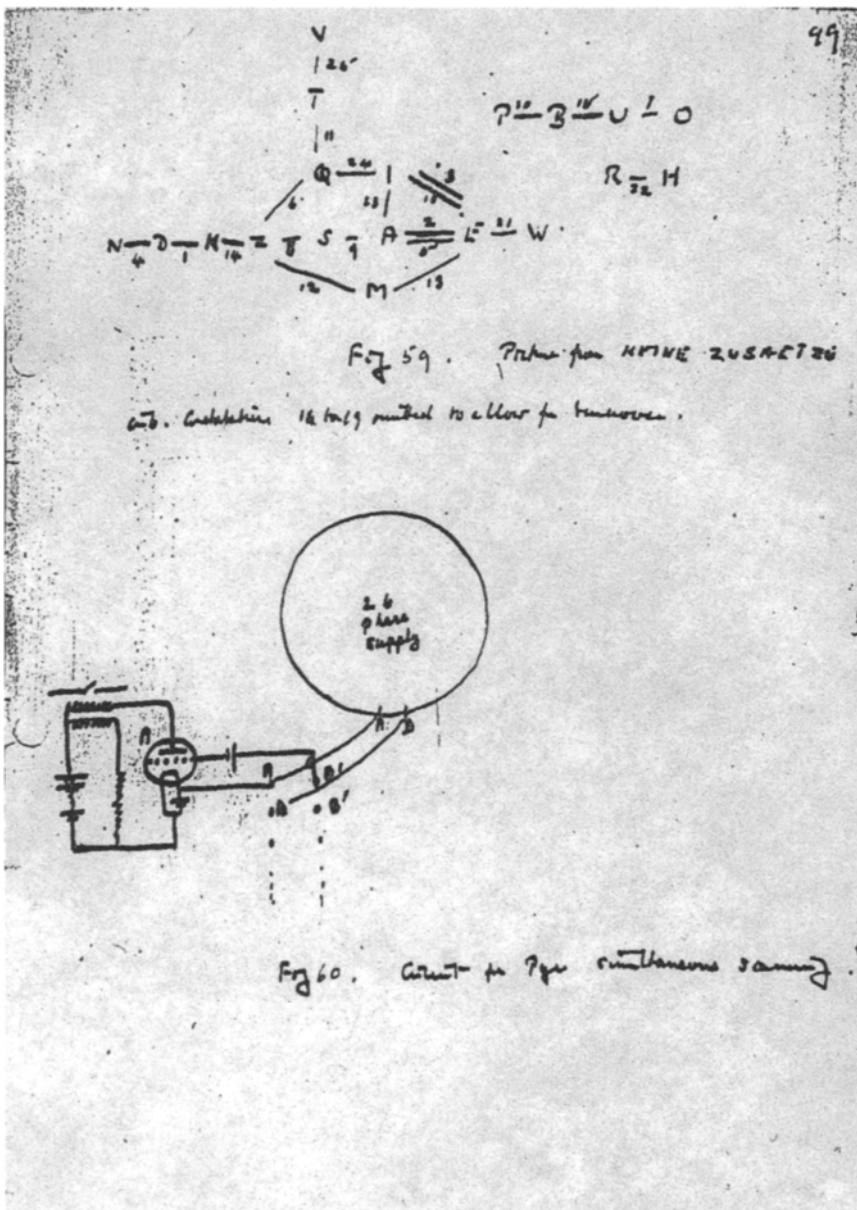
Page 99 shows (in Fig. 59) his diagram of the logical chain of implications that could be deduced from the piece of guessed plaintext as on page 97, and (in Fig. 60) the basic idea of a Bombe to exploit such implications.

The closed cycles in Turing's diagram, for instance Z-S-A-E-M-Z, correspond to chains of logical implications which yield consistency conditions independent of the plugboard.

They make it possible to reject a rotor position, as inconsistent with the ciphertext and plaintext data, even though the plugboard remains unknown.

When these chains of logical implications are exploited, almost every 'wrong' rotor position will be rejected, leaving just a few to be tested in detail. Among them, if the plaintext has been guessed correctly, will be the correct rotor setting.

Turing's hazy sketch of the Bombe does not however explain the crucial ideas that he and G.W. Welchman later used to mechanise the process of following chains of logical implications to the full; these are described in the following pages.



Excerpt 3

Page 104 describes the critical problem of how to ‘scan’ the electrical output from the Bombe to detect the possibility of a correct rotor position. Turing first describes “serial scanning,” in which each of the 26 different plugboard hypotheses (“each Stecker value”) has to be tried in turn, and then raises the possibility of “simultaneous scanning.”

Registering

It now only remains to find a mechanical method of ~~remembering~~
 whether the multiple encipherment condition is fulfilled.
 This can be done most simply if we are willing to test each
 Stecker value of the central letter θ roughout all rod starts
 before trying the next Stecker value. ~~xxxxxxxxxxxxxx~~
~~xxxxxxxxxxxxxx~~
 Suppose we are investigating the case θ where the Stecker value
 of the central letter is K in K . We let α current enter $\alpha 1$ of
 the chains of enigmas at their K input points, and at the K
 output points of the θ chain we put relays. The 'on' points of
 the five relay are put in series with a battery (say), and
~~xxxxxxxxxxxxxx~~
 another relay. A current flows through this last
 relay if and only if a current flows through all the other five
 relays, i.e. if the five multiple encipherments applied to K all
 give K . When this happens the effect is, essentially, to stop the
 machine, and such an occurrence is known at Letchworth as a
 'straight'. An alternative possibility is to have a quickly
 rotating 'sconer' which, during a revolution, would first
 connect the inputs of the θ chain to the current supply, and the
 output points A to the relays, and then would connect the
 input and output points B to the supply and relays. In a
 revolution of the sconer the output and input points A to B
 would all have their turn, and the right hand whirls would
 then move on. This last possible solution was called 'serial
 scanning' and led to all the possible forms of re-introduction
 being known as different kinds of 'scanning'. The simple possibility
 that we first mentioned was called 'single line scanning'.
 Naturally there are much more relays into possible alternatives to
 those two kinds of scanning, which would not enable all the
 possible Stecker values of the central letter to be tested
 at simultaneously without the use of the machine moving.
 Any device to do this was described as 'simultaneous scanning'.

Excerpt 4

Page 107 describes how ‘simultaneous scanning’ can give rise to the ‘very useful’ principle that ‘all the deductions drawn will then be false, and those that remain will stand out clearly as possible correct hypotheses.’

107

When we look at the Bombe in this way we see that it would be natural to modify it so as to make this idea fit even better. We have not so far allowed for lengthy chains of deductions; the possible deductions stop as soon as one comes back to the central letter. There is however no reason why, when from one Stecker value of a hypothesis about the central letter we have deduced that the central letter must have another Stecker value, we should not go on and draw further conclusions from this second Stecker value. At first sight this seems quite useless, but, as all the deductions are reversible, it is actually very useful, for all the conclusions that can be drawn will then be false, and those that remain will stand out clearly as possible correct hypotheses. In order that all these deductions may be made mechanically we shall have to connect the 26 contacts at the end of each chain to the common beginning of all the chains. With this arrangement we can think of each ~~xxxxxxxx~~ output or input point as representing a possible Stecker, and know if two of these points are connected together through the enigmas then the corresponding Stecker imply one another. At this point we might see how it all works out in the case of the crib given above. This crib was actually enciphered with ~~xx..~~

2 3 5 10 25
107

2	3	5	10	25
XH	XH	MD	TD	LV
AP	BU	JZ	LI	WO
QK	EN	CV	MU	DI
CV	PK	HA	XE	OM
TY	QI	YE	OV	XU
UG	AW	GR	JY	FT
MG	OV	PQ	DF	JP
BD	JY	NW	SL	OK
IW	DM	LH	ON	AY
JZ	QZ	IX	BN	NB
GR	SL	FU	AZ	IE
YE	GT	OI	PK	ZG
IL	FC	KT	GM	RK

[237]

Excerpt 5

Page 112 describes Welchman's idea for exploiting the self-inverse property of the Enigma by using the diagonal board.

42
Figs 43, 44 show the connections of enigmas and diagonal board in a particular case. The case of a six-letter alphabet has been taken to reduce the size of the figure.

The actual origin of the spider was not ~~not~~ an attempt to find simultaneous scanning for the Bombe, but in to make use of the reciprocal character of the Stecker. This occurred at the time when it was clear that very much shorter crib would have to be worked than could be managed on the Bombe. Welchmann then discovered that ~~xxxxxxxxxxxxxx~~ by using a diagonal board one could get the complete set of consequences of a hypothesis. The ideal machine that Welchmann was ~~impossible~~ aiming at was to reject any position in which a certain fixed-for-the-time Stecker hypothesis led to any direct contradiction: by a direct contradiction I do not mean to include any contradictions which can only be obtained by considering all Stecker values of some letter independently and showing each one ~~impossible~~ inconsistent with the original hypothesis. Actually the spider does more than this in one way and less in another. It is not restricted to dealing with one Stecker hypothesis at a time, and it does not find all direct contradictions.

The Spider

Naturally enough Welchmann and Keen set to work to find some way of adapting the spider so as to detect all direct contradictions. The result of this research is described in the next section. Before we can leave the spider however we should ~~know~~ see what sort of contradictions it will detect, and about how many atoms one will get with given data.

First of all let us simplify the problem and consider a only ~~xxxxxxxxxx~~ normal stops, i.e. positions at which by altering the point at which the current enters the diagonal board ~~the current~~ ^{the current} ~~can make~~ ^{can make} the relays closed. The current will then be

Excerpt 6

Pages 136–7 describe the context in which Turing made his successful deduction of the bigram key-system in the naval Enigma.

find the keys for other days at the beginning of May and they actually found the Stecker for ~~xxxxxx~~ the 2nd, 3rd, 4th, 5th and 6th, and read about 100 messages. ~~xxxxxxxxxxxxxxxxxxxxxx~~
~~xxxxxx~~ The indicators and window positions of four (selected) messages for the 5th were

Indicator	Window start
KFJK EWTF	P C V
SYLG EWUF	B Z V
JMHO UVQG	M E M
JMFE FEVC	MYX K

The repetition of the EW combined with the repetition of V suggests that the ~~xxxxxxxxxxxxxx~~ fifth and sixth letters describe the third letter of the window position, and similarly one is led to believe that the first two letters of the indicator represent the first letter of the window position, and that the third and fourth represent the second. ~~xxxxxxxxxxxxxx~~ Presumably this effect is somehow produced by means of a table of bigramme equivalents of letters, but it cannot be done simply by replacing the letters of the window position with one of the bigramme equivalents, and then cutting in a dummy bigramme, for in this case the window position corresponding to JMFE FEVC would have to be say MYX instead of MYK. Probably some encipherment is involved somewhere. The two most natural alternatives are . if i) The letters of the window position are replaced by some bigramme equivalents and then the whole enciphered at some 'Grundstellung', or ii) The window position is enciphered at the Grundstellung, and the resulting letters replaced by bigramme equivalents. The second of these alternatives was made far more probable by the following indicators occurring on the 2nd May

KDIP IVJO	V C P
XOKX JXJY	V U E
ROOK JLWA	N U M

With this second alternative we can easily deduce from the

first two indicators that the bigrammes XI and XX have the same value, and this is confirmed from the second and third, where XX and XI occur in the second position instead of the first.

It so happened that the change of indicating system had not been very well made, and a certain torpedo boat, with the call sign AFA: had not been provided with the bigramme tables. This boat sent a message in another cipher explaining this on the 1st May, and it was arranged that traffic with AFA: was to take place according to the old system until May 4, when the bigramme tables would be supplied. Sufficient traffic passed on May 2,3 from to and from AFA: for the Grundstellung used to be found, the Stecker having already been found from the FORTYWEKFY messages. It was natural to assume that the Grundstellung used by AFA: was the Grundstellung to be used with the correct method of indication, and as soon as we noticed the two indicators mentioned above we tried this out and found it to be the case.

There actually turned out to be some more complications. There were two Grundstellungen instead of one. One of them was called the Allgemeine and the other the Offiziere Grundstellung. This made it extremely difficult to find either Grundstellung. The Boles pointed out another possibility, viz that the trigrammes were still probably not chosen at random. They suggested that probably the window positions enciphered at the Grundstellung, rather than the window positions themselves were taken off the restricted list.

In Nov. 1939 a prisoner told us that the German Navy had now given up writing numbers with Y...YY...Y and that the digits of the numbers were spelt out in full. When we heard this we examined the messages on toward the end of 1937 which were expected to be continuous. In a wrote the expected beginnning under them. The appearance of 'crashes' i.e. of letters apparently left unaltered by encipherment, then shows how nearly correct our guesses were. Assuming that the change XXX

Turing's Papers on Programming

PREFACE (by Martin Campbell-Kelly)

With the exception of his early foray into program verification “Checking a Large Routine” [Turing 1949], Turing’s programming papers are an intellectual disappointment. In September 1948 he became Assistant Director of the Manchester University Computing Machine Laboratory, with the academic rank of reader. He was explicitly charged with making the Manchester computer available to a broad class of users and hence for providing the necessary programming infrastructure. Unfortunately for the users, Turing was much more interested in programming computers for his own researches than he was in developing programming systems to make computers more accessible for others.

Prior to joining Manchester University, during 1946–47 Turing had worked with J.M. Wilkinson at the National Physical Laboratory on coding problems for the ACE. However, this was purely programming on paper for a machine that did not yet exist [Wilkinson 1948]. At Manchester, Turing had his first experience of programming a real computer, and he was by far the heaviest user of the prototype machine that was then coming into service. In the “very little time that was available for full scale problems” Turing’s major programming achievements included the testing of Mersenne numbers for primality, and an investigation of the Riemann hypothesis described in his paper “Some Calculations of the Riemann Zeta-Function” (Turing 1953; 1951a, pp. 96–97).

A fully engineered version of the Manchester computer, built by the local electrical engineering firm Ferranti, was installed in the Computing Machine Laboratory at Manchester University in February 1951 – see Fig. 1. (This machine was sold as the Ferranti Mark I, but was – confusingly – known locally as the Manchester Electronic Computer Mark II. Later it was simply known as the Mark I.)

Turing wrote the handbook for this machine, which was printed about March 1951. The handbook consists of some one hundred pages of foolscap typescript and contains much arcane detail of the instruction code of the Mark I – which owed very little to Turing – and has therefore not been included in the *Collected Works*. However, there is a fascinating section in the handbook, “Programming Principles,” which is pure Turing and is reproduced here. The extract is full of the practical programming savvy that Turing had acquired through struggling with the prototype machine. He describes the use of subroutines, the use of flow diagrams (which he calls “block schematic diagrams”), and “check sheets” which he used to compare the predicted behaviour of a program with its real execution as an aid to debugging. He explains what was later called the space-time trade off,



Fig. 1. Console of the Manchester Mark I computer, c.1951. Alan Turing is standing at right.

and also describes program decomposition techniques that later went under such names as step-wise refinement and bottom-up coding. Fascinating as this material is, it is not in any way an important contribution to the science of programming; rather it is typical of the programming practices that were independently arrived at in many of the early computer laboratories in Britain and the United States in the early 1950s.

The least satisfactory aspect of Turing's work with the Mark I was the programming notation with which he saddled users of the machine. Fig. 2 is a reproduction of Turing's INPUT routine, which was used to read programs into the machine from punched paper-tape – a primitive form of what was later called an assembler. It would only be possible to understand this piece of code by a thorough study of the full programmer's handbook, and even then it would not be straightforward. It is reproduced here essentially as an exhibit – it tells us what Turing's programs “looked like.” Each program instruction consisted of 20 bits, which Turing wrote down as four characters using the 5-bit Post Office teleprinter code. In effect he used the teleprinter code as a base-32 number system of which the digits were:

/E@A:SIU₂¹DRJNFCKTZLWHYPQOBG"MXV£

To compound the notational difficulty, numbers were written backwards, so that the sign was at the rightmost end (this being the direction in which numbers were displayed on a serial machine such as the Mark I). Because zero was represented by the forward-stroke character, and this was the most commonly-used character in the written form of programs and data, one early user decided this must be an unconscious reflection of the famously dismal Manchester weather as the effect was that of rain seen through a dirty window pane! None of Turing's programme sheets provided annotations to explain the coding, leaving users to either accept them at face value or to laboriously decipher the code.¹

Turing was seemingly oblivious of the mental gymnastics his notation required of ordinary users. Maurice Wilkes, who developed a much cleaner and more user-friendly notation for the contemporary EDSAC at Cambridge University, subsequently remarked that the notation was "bizarre in the extreme", adding further that Turing "had a very nimble brain himself and saw no need to make concessions to those less well endowed" (Campbell-Kelly 1980). Thus whereas Cambridge computer users would write *A 250 L* for an instruction such as *add the long number in location 250 into the accumulator*, Manchester programmers would have to write something like *GUTC*. (Here the addition operation code is represented by *TC* while the address 250 is written as *GU* in base-32 backwards).

Turing's only formal publication on programming techniques for the Manchester Mark I, "Local Programming Methods and Conventions", was read at the Inaugural Conference for the Manchester University Computer in July 1951 (AMT 1951b) – see Fig. 3.

This paper, which is rather trivial in content, seeks less than successfully to justify the choice of the awkward base-32 notation for both instructions and numbers. He comments, for example, that "Since the names of store lines are little more than arbitrary labels it matters little what code is used for them." Surely only Turing could have believed that *GU* and 250 were equally good names for a storage location.

In defence of Turing's programming system, however, one should note that – despite the awkward notion – the environment he created was business-like and well organised. Thus library subroutines were tested and documented in a standard way, and Turing devised a systematic operating regimen for using the machine. It is likely that this attention to practicalities was a cultural inheritance from his wartime experiences at Bletchley Park. Finally, there was one unexpected benefit of Turing's programming notation: users found it so uncomfortable

¹ The example reproduced in Fig. 2 bears the annotations of Christopher Strachey – an early user of the machine.

MANCHESTER UNIVERSITY COMPUTING MACHINE LABORATORY.

PROGRAMME SHEET I.

<u>Name of Routine.</u> <i>INPUT.</i>	<u>Date.</u> 1.7.51.
<u>Purpose.</u> To read from tapes.	
<u>Ques.</u> <i>J @ L V A / @ / .</i>	
<u>Sub-routines.</u>	<u>Principal Lines.</u> $[\ / E]_0^{19} = A / @ /$ $[\ / A]_0^{19} = A E K /$ on entry and after " $[\ / A]_0^{19} = R / /$: otherwise
<u>Tapes.</u> INPUT ONE SPECIAL. INPUT TWO SPECIAL. INPUT THREE SPECIAL. INPUT FOUR SPECIAL	<u>Magnetic Storage.</u> 2 L & R, 3 L & R. <u>Electronic Storage.</u> S.O. S.I.
<u>Stores Altered.</u> <i>B 4, B 5, B 6.</i>	<i>BK</i>
<u>Effects.</u> Enter either (1) By cue, or (2) By setting $[H] = A / / /$ and operating K E C. (3) Through INITIAL by operating K E C. <i>With warning characters see the Programmers' Handbook pp. 34-36.</i>	
<p><i>BK is B K and for warning characters see the Programmers' Handbook pp. 34-36.</i></p> <p><i>XVKHO sets B3 (VK), and 34 as the word length.</i></p> <p><i>34 is stored with warning characters K, Z, W, Y, Q, X . . . and T, " .</i></p> <p><i>+</i></p>	

Fig. 2.

MANCHESTER UNIVERSITY ELECTRONIC COMPUTER.

Programme Sheet 2.

ROUTINE INPUT (SHEET 1).

1	/ E / / / A / @ / E	2	R / / : / A E K / A
/ E / / E / / T : E	/ V K / / A E		
/ E / / @ T E / J	/ Y : / C @ A E		
/ E / / A E / P	/ V K / / N A A A E		
/ E / / : S K T B	/ V K / / A : A E		
/ E / / S V K T /	355-22 / V K H O O S A E		
/ E / / I Z E / C	/ D S Y O I U A A E		
/ E / / U M K / N	/ D C @ / P F D A A E		
/ E / / 2 M K T A	/ B @ / P F R L @		
/ E / / : D R / / :	/ M M @ / P R D A A E		
/ E / / R V K / E	/ U U @ / J L L @		
22 Y @ / : J J : / C	/ @ / E @ J L L @		
/ E E T : N V K / N	/ V S T / A P J N L L @		
/ E E T £ F V K / A	/ / / P F C L L @		
/ E E T £ G V K P O	36-22 / M K T 2 F C K L L @		
24 X @ / : K / A I P	/ J S Y G T K T Z L L @		
/ E E T £ T / X £ /	/ K S / H A P L L L @		
L E E T £ Z R / / /	/ / / H A P L L L @		
/ E E T £ L / / /	/ / / H A P L L L @		
25 C F S E T £ W H K T /	36-22 / J S Y O P W H L L @		
/ E E T £ H N S / P	/ / / A / Y L F H L L @		
O E E T £ Y / 2 Y B	36-22 / E / / : Y P F H L L @		
/ E E T £ P / D H B	/ J @ / / : P Q O L L L @		
26 Q @ T £ Q J @ L V	/ / / A / : Q Q O B G L L L @		
/ E E T £ Q O A / @ /	/ J @ / / : Q Q E E T £ B G L L L @		
/ E E T £ B Q E T /	36-22 / Q Q @ / / : Q Q E E T £ B G L L L @		
/ E E T £ G N S / P	/ J @ / / : Q Q E E T £ B G L L L @		
B @ T £ " 2 E / E	/ G G @ / / : M M 2 E		
/ E E T £ M R / / :	36-22 / J @ / / : E E K X L L @		
" @ T £ X V K / A	/ E E K X L L @		
/ E E T £ V V : / C	36-22 / Q @ / / : E E V L L @		
(36) / E T : £ V K / N	/ / / : E E F / E		

Tape:- INPUT ONE SPECIAL

INPUT TWO SPECIAL.

Fig. 2. (continued).

MANCHESTER UNIVERSITY ELECTRONIC COMPUTER.

Programme Sheet 2.

ROUTINE. INPUT (SHEET 2).

Tape:- INPUT THREE SPECIAL

INPUT FOUR SPECIAL

LOCAL PROGRAMMING METHODS AND CONVENTIONS

By A. M. Turing, B.A., Ph.D., F.R.S.

In this talk I am speaking about the use of teleprint code for writing down instructions and other material stored in the machine, and also about the conventions whereby rows of digits represent numbers.

In connection with the use of the teleprint code, it is necessary to choose between doing one's programming in a binary form, or else using some other form and allowing the machine to convert to the binary form. Binary scale working can be replaced by the use of any power of two as radix, and the choice made at Manchester was the scale of 32. It is probable that a scale of 8 or 16 would be slightly more convenient, but there is little to choose. The following points are made in favour of the teleprint (scale 32) method, as opposed to a decimal method.

(i) Since the instructions are held in the machine in a form intelligible to the programmer if it is very easy to follow what is going on by looking at the monitor tubes. I express scepticism over the view that this so-called 'peeping' can or should be entirely eliminated by the use of automatic checking programmes. I would prefer to bring the peeping procedure to such efficiency that one's errors could be found extremely quickly by it. As practised at Manchester this procedure consists of the preparation in advance of 'check-sheets' purporting to describe the behaviour of the machine in detail. The behaviour of the machine can then be compared with these check-sheets. Peeping could, however, be objectionable unless, when an error is found, the programmer leaves the machine to someone else whilst he thinks about it.

(ii) Since the names of the store lines are little more than arbitrary labels it matters little what code is used for them. The function part of an instruction can in any case only be a quite arbitrary symbol.

(iii) In the case of the Manchester machine there are additional reasons. There is a natural division of the store into tubes, each of 64 lines. This division cannot be ignored because transfers of information between the electronic and magnetic stores is always by complete tube-fulls. Each tube consists of two 'columns' each of 32 lines, so that each line can be described by two teleprint characters, one for the column and the other for the position in the column.

In connection with the relation between rows of digits and numbers, the nature of the multiplier makes it inconvenient

to use only one convention by which rows of digits represent numbers. Four conventions are used at Manchester, which can be illustrated in connection with five digit rows, e.g., 00011. With all of these conventions as well as others such as the Cambridge convention, the numbers concerned form an arithmetic progression. The row 00000 always represents the number 0, and rows representing successive terms of the progression are related by 'row addition' of 10000. The most significant digit is on the right. Under these circumstances the convention is completely determined by giving the range of numbers which can be represented by it. The four Manchester conventions and the Cambridge convention can then be described by the table:

	Least Number	Greatest Number	Value for 00011	Suffix
Plus	0	31	24	+
Plus or Minus	-16	15	8	\pm
Plus Fractional	0	31	3	$\pm f$
Plus or Minus	1	15	1	
Fractional	2	32	4	$\pm f$
Cambridge	-2	$\frac{15}{8}$	-1	C (suggested)

If the suffix shown in the last column is combined with a symbol representing a row, then the resulting expression represents the corresponding number. Thus $(00011)\pm$ is synonymous with \pm . It is desirable also to have a notation by which numbers are transformed into rows of digits. If α is a number then $[\alpha]$ represents that part of the binary expression of α which runs from the coefficient of 2^n to that of 2^0 . Thus for instance $[3:5]_2$ is 01100.

The provision of such notations makes it possible to describe operations occurring in the machine concisely and unambiguously. Thus for instance the operation of 'copying a store line into the accumulator with extension of the sign digit throughout the most significant half of the accumulator' can be represented by the equation $A' = [S \pm]$. The convention that undashed letters represent values before the operation, dashed letters values after it, is always used.

DISCUSSION CONTRIBUTIONS

Mr. T. J. Rey

It has been asked if programming can be simplified. The answer depends on whether one refers to the initial or the final steps of human labour. To the extent that a calculating machine is useful because it can perform many operations for only a few instructions, a working knowledge of the specialised notations involved is essential. However, as regards the instruction code, its details can be built into specialised mechanisms. For example, a key may be labelled 'x' whereas its business end will punch that group of binary digits which eventually sets up the multiplication circuit. Similarly, in the Mark VI B.T.L. Computer, instructions may be set up by threading wires through a matrix array of toroidal transformer cores; the operation

$A \times B - C$ (Next Instruction ?)

is then prepared on passing a wire through one core in each column, successive cores being labelled A , \times , B , C , 7 ; a pulse on this wire energises the transformers so as to stimulate control circuits via the secondary windings.

Mr. E. A. Newman

Mr. Turing suggests that all computation consists essentially of multiplication. This is not so. As an example, take the extraction of the roots of a polynomial by Horner's Method. Some computers have estimated that on the bulk of problems multiplication time in a machine would have to be of the order of 100 times addition or transfer time before it appreciably slowed the overall computing speed. There are, of course, exceptional problems for which this is not so.

Fig. 3.

that Manchester University quickly came to the forefront in developing automatic coding systems to obviate the need for hand coding. Other computer laboratories, with more user-friendly notations, took much longer to take this step.

References

- Turing, A.M. 1949. "Checking a Large Routine", *Report of a Conference on High Speed Automatic Calculating Machines*, 22–25 June, pp. 67–8. Reprinted with a commentary as F.L. Morris and C.B. Jones, "Early Program Proof by Alan Turing", *Annals of the History of Computing*, Vol. 6, April 1984, pp. 139–43.
- Turing, A.M. 1951a. *Programmers' Handbook for the Manchester Electronic Computer Mark II*, Manchester University Computing Machine Laboratory.
- Turing, A.M. 1951b. "Local Programming Methods and Conventions", *Manchester University Computer Inaugural Conference*, p. 12.
- Turing, A.M. 1953. "Some Calculations of the Riemann Zeta-Function", Collected Works, *Pure Mathematics*, pp. 79–97.
- Campbell-Kelly, M. 1980. "Programming the Mark I: Early Programming Activity at the University of Manchester", *Annals of the History of Computing*, Vol. 2, 130–168.
- Wilkinson, J.W. 1948. *Progress Report on the Automatic Computing Engine*, NPL Report Ma/17/1024.

Excerpt from: Programmer's Handbook for the Manchester Electronic Computer Mark II

Programming Principles

Programming is a skill best acquired by practice and example rather than from books. The remarks given here are therefore quite inadequate.

If it is desired to give a definition of programming, one might say that it is an activity by which a digital computer is made to do a man's will, by expressing this will suitably on punched tapes, or whatever other input medium is accepted by the machine. This is normally achieved by working up from relatively simple requirements to more complex ones. Thus for instance if it is desired to do a Fourier analysis on the machine it would be as well to arrange first that one can calculate cosines on it. For each process that one wishes the machine to be able to carry out one constructs a 'routine' for the process. This consists mainly of a set of instructions, which are obeyed by the machine in the process. It is not usual however to reckon all the instructions obeyed in the process as part of the 'routine'. Many of them will belong to other routines, previously constructed and carrying out simpler processes. Thus for instance the Fourier analysis process would involve obeying instructions in the routine for forming cosines as well as ones in the analysis routine proper. In a case like this the cosine's routine is described as a 'subroutine' of the analysis routine. The subroutines of any routine may themselves have subroutines. This is like the case of the bigger and lesser fleas. I am not sure of the exact meaning the poet attached to the phrase 'and so on ad infinitum', but am inclined to think that he meant there was no limit that one could assign to a parasitic chain of fleas, rather than that he believed in infinitely long chains. This certainly is the case with subroutines. One always eventually comes down to a routine without subroutines.

What is normally required of a routine is that a certain function of the state of the machine shall be calculated and stored in a given place, the majority of the content of the store being unaffected by the process, and the routine not being dependent on this part having any particular content. It is usual also for the other part of the store to be divided into a part which is altered in the process but not greatly restricted as to its original content, and a part which is unaltered in its content throughout, and such that the correct working of the routine depends on this part having a particular content. The former can be described as 'the working space for the routine' and the latter as 'the space occupied by the routine'.

Applying this to the Mark II machine, the routines usually 'occupy' various tracks of the magnetic store. The working space includes all the electronic store,

with the exception of PERM², which can equally well be reckoned as working space which is never really altered, or as a part common to all routines. The first two pages of the electronic store are also somewhat exceptional, if normal conventions are used, as they are only altered when one is copying new routines from the magnetic store onto them.

These definitions do not really help the beginner. Something more specific is needed. I describe below the principal steps which I use in programming, in the hope they will be of some small assistance.

i) **Make a plan.** This rather baffling piece of advice is often offered in identical words to the beginner in chess. Likewise the writer of a short story is advised to 'think of a plot' or an inventor to 'have an idea'. These things are not the kind that we try to make rules about. In this case however some assistance can be given, by describing the decisions that go to make up the plan.

- a) If it is a genuine numerical computation that is involved (rather than, e.g. the solution of a puzzle) one must decide what mathematical formulae are to be used. For example if one were calculating the Bessel function $J_0(x)$ one would have, amongst others, the alternatives of using the power series in x , various other power series with other origins, interpolation from a table, various definite integrals, integration of the differential equation by small arcs, and asymptotic formulae. It may be necessary to give some small consideration to a number of the alternative methods.
- b) Some idea should be formed as to the supply and demand of the various factors involved. A balance must always be struck between the following incompatible desires:

- To carry the process through as fast as possible
- To use as little storage space as possible
- To finish the programming as quickly as possible
- To achieve the maximum possible accuracy

We may express this by saying that machine time, storage space, programmer's time and inaccuracy of results all cost something. The plan should take this into account to some extent, though a true optimum cannot be achieved except by chance, since programmer's time is involved, so that a determination of the optimum would defeat its own ends. The 'state of the market' for these economic factors will vary greatly from problem to problem. For instance there will be an enormous proportion of problems (40% perhaps) where there is no question of using the whole storage capacity of the machine, so that space is almost free. With other types of problem one could easily use ten million digits of storage and still not be satisfied.

² System code that was permanently kept in the electronic store; the space occupied was not therefore generally accessible to the user.

The space shortage applies mainly to working space rather than to space occupied by the routines. Since these usually have to be written down by someone this in itself has a limiting effect. Speed will usually be a factor worth consideration, though there are many 'fiddling' jobs where it is almost irrelevant. For instance the calculation of tabular values for functions, which are to be stored in the machine and later used for interpolation, would usually be in this class. Programmer's time will usually be the main factor in special jobs, but is relatively unimportant in fundamental routines which are used in most jobs. Accuracy may compete with machine time, e.g. over such questions as the number of terms to be taken in a series, and with space over the question as to whether 20 or 40 digits of a number should be stored.

- c) The available storage space must be apportioned to various duties. This will apply both to magnetic and electronic storage. The magnetic storage will probably be mainly either working space or unused. It should be possible to estimate the space occupied by instructions to within say two tracks, for a large part will probably be previously constructed programmes, occupying a known number of tracks. The quantities to be held in the working space should if possible be arranged in packets which it is convenient to use all at once, and which can be packed into a track or a half-track or quarter-track. For instance when multiplying matrices it might be convenient to partition the matrices into four-rowed or eight-rowed square matrices and keep each either in a track or a quarter-track. The apportionment of the electronic store is partly ruled by the conventions we have introduced, but there is still a good deal of freedom, e.g. if eight pages are available then pages 4, 5, 6 can be used for systematic working space and may be used for various different purposes that require systematic working space.

The beginner will do well to ask for advice concerning plans. Bad plans lead to programmes being thrown away, wasting valuable programmer's time.

- d) If questions of time are at all critical the 'plan' should include a little detailed programming, i.e. the writing down of a few instructions. It should be fairly evident which operations are likely to consume most of the time, and help decide whether the plan as a whole is satisfactory. Very often the 'omission of counting method' should be applied.
- e) If one cannot think of any way, good or bad, for doing a job, it is a good thing to try and think how one would do it oneself with pencil and paper. If one can think of such a method it can usually be translated into a method which could be applied to the machine.

(ii) Break the problem down. This in effect means to decide which parts of the problem should be made into definite subroutines. The purpose of this is

partly to make the problem easier to think about, the remaining work consisting of a number of 'limited objective' problems. Another reason for breaking down is to facilitate the solution of other problems by the provision of useful subroutines. For instance if the problem on hand were the calculation of Bessel functions and it had already been decided to use the differential equation, it might be wise to make and use a subroutine for the solution of linear second order differential equations. This subroutine would in general be used in connection with other subroutines which calculate the coefficients of the equation.

(iii) Do the programming of the new subroutines. It is better to do the programming of the subroutines before that of the main routine, because there will be a number of details which will only be known after the subroutine has been made, e.g. scale factors applied to the results, number of pages occupied by the subroutines, etc. It also frequently happens in the making of the subroutine that some relatively small change in its proposed properties is desirable. Changes of these details may put the main routine seriously out if it were made first. There is a danger that this procedure may result in one's 'not seeing the wood for the trees', but this should not happen if the original plan was well thought out. The programming of each subroutine can itself be divided into parts:

- a) As with programming a whole problem a plan is needed for a subroutine. A convenient aid in this is the 'block schematic diagram'. This consists of a number of operations described in English (or any private notation that the programmer prefers) and joined by arrows. Two arrows may leave a point where a test occurs, or more if a variable control transfer number is used. Notes may also be made showing what is tested, or how many times a loop is to be traversed.
- b) The operations appearing as blocks in a) may be replaced by actual instructions. It is usually not worth while at first to write down more than the last two characters of the (presumptive) instruction, i.e. the B line and function parts. These are quite enough to remind one of what was the purpose of the instruction.
- c) One may then write the instructions into a page, deciding at the same time what are to be the addresses involved.
- d) When the programming is complete, check sheets must be made. It is often advisable to start making check sheets long before the programme is complete; one should in fact begin them as soon as one feels that one has got into a muddle. It is often possible to work out most of the programme on the check sheets and afterwards transfer back onto the page or pages of instructions.

(iv) Programme the main routine. This follows principles similar to (iii).

Of course these remarks merely represent one possible way of doing programming. Individuals will no doubt vary as to the methods they prefer.

Minimum Cost Sequential Analysis

(An unpublished manuscript located in King's College, Cambridge
– comprising 6 typed pages, 7 handwritten pages and 3 figures)

Excerpt: Page 1 only – a summary

Barnard has described a system of 'sequential analysis' for deciding whether or not to reject a batch of articles because too large a proportion are defective. In this method articles are taken from the batch at random and tested. As the testing proceeds running totals of the numbers n_D of 'defectives' and n_E of 'effectives' are kept. The points (n_D, n_E) may be related on a graph, to form a crooked line. The plane may be divided into three sections according to the three possible procedures to be applied. One may either accept, reject or continue to test. We therefore have 'accept', 'reject' or 'continue' regions. Naturally, the interior points of the accept, reject or continue regions cannot (except by oversight) actually come into play, since the testing ceases as soon as the point reaches the boundary of the test region. The procedure is therefore completely described by giving the test-accept and test-reject boundaries. They are normally known more shortly as the acceptance and rejection boundaries. Barnard shows that certain straight boundaries are appropriate if certain assumptions are made concerning the purpose which the test is required to fulfill. In this paper I shall make rather different assumptions, and obtain a different form of boundary. Roughly speaking the tests are required to be the cost-economic ones, taking into account the financial value of the loss of goodwill in selling batches which contain too many defectives, the waste involved in unnecessary rejection of batches, and the cost of the testing itself. It will be shown that the test must be of the type described above, and the methods for calculating the acceptance and rejection boundaries will be described.

Commentary (by I.J. Good)

Turing does not give an explicit citation but he refers to Barnard's use of straight-line boundaries for the acceptance or rejection of a batch of items. Barnard [1946, Fig. 2] shows such boundaries so probably this is the reference that Turing had in mind. Also, Turing uses the notation p , as did Barnard, for the proportion of defective items in the batch.

On the second page of Turing's paper he gives examples to show that it is necessary to assume a prior distribution for p . He calls it an *a priori* distribution as

was common fifty years ago. Barnard [1946], only reluctantly, mentions Bayes explicitly, after about 20,000 words, in the last paragraph of his *Conclusion* “designed to provoke controversy”. He says “in the general problem of sampling inspection we have an example of a practical situation in which we not only can, but must, apply Bayes’ theorem”. Again, in Barnard [1947] he actually avoids the dirty five-letter word. By his emphasis on likelihood, Barnard might at first have given Turing the impression of being against Bayesianism. This might have made Turing think it was worthwhile to bring Bayes in by the front door, whereas Barnard [1946] admitted Bayes only by the back door.

Turing says that the solution to the problem of minimum cost sequential analysis will not depend very greatly on the form of the *a priori* distribution of p . In modern terminology he is asserting *robustness* or *insensitivity* of the procedure with respect to the prior.

After describing the basic logic entirely clearly, Turing launches into the nitty-gritty indigestible algebra which is appropriate for the Turing archives.

Historical Note. Turing was one of the first, in 1940 or 1941, to propose a Bayesian sequential procedure, and for a very important application. His method went into operation in the Spring of 1941 (until mid-1943) under the name *Banburismus*. This was a cryptanalytic procedure used against the German Naval Enigma in World War II. It made use of what is now called the Bayes factor method. The logarithm of a Bayes factor is sometimes called “weight of evidence”. For more details and references, going back to 1878, see Good [1992]. In that article I said, referring to a very brief private meeting in 1941 or 1942 (more probably in 1941), “Barnard said that curiously enough a similar method [similar to the Bayesian sequential procedure which I mentioned, of course without mentioning the application] was being used for quality control in the Ministry of Supply for discriminating between [good and bad] lots [batches of items] rather than [between] hypotheses.” (Barnard forgot about the meeting but I recall it with great clarity and with several further details.) But I must have been wrong in Good [1992] when I said Barnard’s method was sequential at the time of our brief meeting. For Barnard [1947] says “At the beginning of 1943, the idea underlying sequential analysis had not yet been formulated”; and in Barnard [1946, p. 2]

“As Wald points out, many authors have used sequential ideas in particular cases, without realising the general implications. His work, done in April 1943, seems to have been the first general attack, while mine was started in June of that year.”

That Turing’s work was “general” is clear especially from his theorems, discovered in connection with sequential analysis, about the expected value and variance of “decibannage” (weight of evidence): see for example, Good [1961, p. 129]. The penultimate sentence of Good [1992] is therefore misleading.

At the time of our meeting, no doubt Barnard was using the likelihood ratio, in the Ministry of Supply, for quality control, but not with the Bayesian interpretation.

It is possible that Turing was provoked to write about sequential analysis because he had previously invented the topic.

There is a passage in Barnard [1947, p. 659] that anticipates a part of the simplest justification of the definition of weight of evidence [Good, 1989a, b] except that he refers to the likelihood ratio without the Bayesian aspect.

Barnard [1954] uses a Bayesian approach, including the Bayes factor approach (see also [Good 1950]). In Barnard's *Conclusion* he says "...not only must we know the prior distribution ... but we may have to know it in considerable detail." In this respect he differs from Turing's opinion. Barnard's paper was presented at a meeting of the Royal Statistical Society on March 17, 1954. One of the discussants was D.G. Champernowne who was a close friend of Turing's and might have informed Turing of the meeting just before Turing died on June 7, 1954. It is clear that Turing didn't know of the meeting when he wrote his unpublished paper on sequential analysis. If he did hear about it, possibly from Champernowne, that would have been a clearly sufficient reason for his not trying to publish his paper in 1954.

References

- Barnard, G.A. [1946] Sequential tests in industrial statistics, *J. Royal Statist. Soc. Suppl.* **8** (1), 1–26, with discussion.
- Barnard, G.A. [1947] Review of *Sequential Analysis* by Abraham Wald (John Wiley, 1947) in *J. Amer. Statist. Assoc.* **41**, 658–664.
- Barnard, G.A. [1954] Sampling inspection and statistical decisions, *J. Royal Statist. Soc. B* **16**, 151–174.
- Good, I.J. [1950] *Probability and the Weighing of Evidence* (Griffin: London).
- Good, I.J. [1961] Weight of evidence, causality and false-alarm probabilities, in: *Information Theory, Fourth London Symposium* (1960) (Colin Cherry, ed; Butterworths: London), 125–136.
- Good, I.J. [1989a] Yet another argument for the explication of weight of evidence, *J. Statist. Comput. & Simul.* **31**, 58–59.
- Good, I.J. [1989b] Weight of evidence and a compelling metaprinciple, *J. Statist. Comput. & Simul.* **31**, 121–123.
- Good, I.J. [1992] Introductory remarks for the article in *Biometrika* 66, "A. M. Turing's statistical work in World War II", in: *Collected Works of A.M. Turing: Pure Mathematics*, 211–223.

This Page Intentionally Left Blank

The Nature of Turing and the Physical World

[A discussion of the letter written by Robin Gandy to Max Newman in June 1954 – the letter follows at the end of the discussion.]

by Andrew Hodges

In this letter, Alan Turing's student, friend and colleague Robin Gandy tried to capture the intellectual currents that seemed most striking in the immediate aftermath of his sudden death. This note attempts to encapsulate the background to Turing's extraordinary exploration of new ideas in physics, as documented by the letter. It also indicates the context of Robin Gandy's knowledge of Turing's work. The reader will know from the Preface to this volume that the Collected Works project itself long rested on Robin Gandy's subsequent dedication, so that Robin's letter was both a last word on Turing's living communications and a first contribution to this necessary future undertaking.

Turing's first serious interest in mathematics came through reading Einstein and Eddington and he had a fine understanding of the principles of general relativity and quantum mechanics while still at school [Hodges 1983; pages 33, 40]. His first recorded serious scientific question was influenced by Eddington's 1928 book, *The Nature of the Physical World*: is there a quantum-mechanical basis to human will? [Hodges 1983; page 63]

As a student, Turing ignored the conventional Cambridge division between 'pure' and 'applied' mathematics. He was seriously reading Russell, but also seriously considered research in mathematical physics [Hodges 1983; page 95] and his Fellowship dissertation on the Central Limit Theorem was on a borderline between 'pure' and 'applied.' John Britton's editorial comment (in *Collected Works of A.M. Turing: Pure Mathematics*) on Turing's lasting interest in applying probability theory extends also to an interest in the physical world quite unlike anything that the Cambridge tradition of pure mathematics encouraged.

Indeed it could be said that Turing treated mathematical logic like an applied mathematician. Max Newman's Memoir [this volume, p. 271] called him 'at heart more of an applied than a pure mathematician' which must have surprised all those who thought they knew Turing as a logician. In 1937, Turing immediately turned to the embodiment of primitive logical operations in electromagnetic relays, and this fascination with connecting the abstract with the practical continued undiminished thereafter, as indeed Robin Gandy's letter pointed out in expressing his agreement with Newman.

One of the most fascinating questions about Turing's subsequent development concerns the extraordinary flux of his ideas in 1937–9, a period when he was busy in logic and analytic number theory, but also tackling the Enigma and arguing with Wittgenstein. At this time he developed his most abstract and advanced work, the ordinal logics. But even this he gave an extra-mathematical interpretation. The ordinal logics are motivated by Gödel's proof that seeing the truth of mathematical statements requires methods which cannot be mechanised. Turing described these non-mechanical steps as 'intuitive judgments'. Had he pursued his interest in an underlying physical viewpoint, what would he have said the brain was doing in such moments of intuition? The question might have driven him to something resembling Penrose's position, as it emerged in Penrose [1988], and later the better known Penrose [1989], on the necessity for an uncomputable element in fundamental physics.

In his Memoir, Max Newman dwelt on the loss to science that arose because this period was terminated by the outbreak of war. Newman's choice of priorities, emphasising the ordinal logics and what Turing might have done had he continued to concentrate on mathematics, but regarding the wartime work and the building of electronic computers as unfortunate or unimportant interruptions, must have struck many as an incomprehensible distortion. In a longer term these judgments may acquire new force.

Leaving aside what Turing might have done, the reality of the Second World War gave immense stimulus to the physical embodiment of Turing's logical ideas, and developed his acquaintance first with electromagnetic and and then with electronic technology. In wartime years, no dividing line was drawn between Turing's logic and its physical implementation. A recently declassified paper of 1942³, Turing's report on his visit to the factory where the American Bombe were built, is striking in its revelation of its author as reporting authoritatively on engineering questions (e.g., the electronic testing of commutators) as well as on logical design.

It was in this context that Turing determined to outsmart American speech-scrambling with his own electronic system, the Delilah. Shortly afterwards, in 1944, Robin Gandy first began close contact with Turing, who was then soldering electronic components with his own hands, and enjoying the role of solving electrical engineering problems (starting, naturally, from Maxwell's equations.) One of many ironies in Gandy's story, which emerged in the many discussions I had with him while writing my Turing biography in 1977–83, was that he then saw himself a mathematical physicist, and so was ignorant of logic. Although in Turing's presence in 1945, the future logician and Turing disciple did not ob-

³ [Turing 1942a]. I owe this to Lee A. Gladwin of the National Archives and Records Administration, Washington DC.

serve the emergence of the practical stored-program computer from the logic of the universal machine.

It is another curious fact that neither Robin Gandy nor anyone else seems to have observed the important change in Turing's position regarding uncomputability and intelligence, which I have recently discussed in [Hodges 1997] as evident by 1945. During the war Turing had apparently come to the position that intelligence did not mean infallibility, so that undecidability and incompleteness were not relevant to understanding mental action. I now think this was a key point in Turing's development, in which he abandoned the significance he had attached to ordinal logics in 1938–9, and instead attached great importance to the ability of computers to modify their own programs and do what programmers could not have foreseen. I believe he decided that 'intuition' could be accounted for by learning processes and the implicit programming of neural networks. But this was apparently a dialogue about logic and physics in Turing's own mind alone.

Turing's knowledge and interest in applied physics did not end with the Second World War. Turing worked from first principles on the design of delay lines for his ACE plan, though the results did not convince professional engineers and his circuit designs were probably his least satisfactory work. In this he failed where his rival Maurice Wilkes, a classic Cambridge applied mathematician, brilliantly succeeded. It should also be said that Turing's knowledge of applied mathematical techniques continued to surprise those who classified him as a pure mathematician. Thus, at Manchester he impressed Alick Glennie, who did computational work for the British atomic bomb, with his current knowledge of hydrodynamics. In his own work, he effortlessly introduced inverse differential operators for handling partial differential equations in his morphogenetic theory.

In the more theoretical side of the physical basis of digital computing, Turing showed rather an inconsistent interest. He maintained a careful concern to point out that logical discrete systems are embodied in the physically continuous, but gave only slight references to actual physics. His 1948 work [Intelligent machinery, *Collected Works of A.M. Turing: Mechanical Intelligence*, p. 111] contains a thermodynamic calculation relevant to computer reliability (see also the story told by John Britton in the introduction to the *Pure Mathematics* volume) but he paid no attention to the quantum-mechanical basis of electronics.

Turing's underlying thesis, increasingly evident as his claims for mechanical intelligence grew in confidence, was that whatever it is the brain does, it must be a computable process. Considering the importance of this conviction, his references to underlying physical law, as discussed in [Hodges 1997], are somewhat thin and cavalier. It is surprising also that although Robin Gandy moved to logic under Turing's influence, and became his student with a thesis on the logical foundations of physics, they never seem to have discussed Turing's underlying assumptions about fundamental physics. Turing never considered quantum com-

putation. Nor did Turing ever press questions about digital approximations to continuous systems. In the famous paper *Computing Machinery and Intelligence* [1950] (republished in *Collected Works of A.M. Turing: Mechanical Intelligence*, p. 133–160) he gave a classic comment (p. 139 or p. 440 of the original) on the ‘butterfly effect’ in physical systems (in term of snowflakes and avalanches), pointing out the lack of analogy in discrete computation; and yet also in brief words (dealing with the ‘Argument from Continuity in the Nervous System’ on p. 151 or p. 451 of the original) espoused faith in the discrete approximations used in applied mathematics. This was entirely consistent with his practical experience: he was pioneering the use of digital computers for exploring the evolution of critical effects in his morphogenetic theory. But it was not a serious examination of the relationship of computation to continuous physics, such as modern theoreticians of analogue computing now undertake.

This somewhat patchy nature of Turing’s post-war physical interests means that there is little to prepare us for a sudden explosion of ideas about the fundamental physics of quantum mechanics and relativity in 1953–4. There is in fact just one link between Turing’s major work in logic and computability, and his late interest in physics. In his radio talk of 1951 [see item 4 in the Appendix, this volume, p. 293], Turing referred briefly to the unpredictability of quantum mechanics as implying that physical systems might not be amenable to simulation by the universal Turing machine. In [Hodges 1983, p. 441] I brought out the reference that Turing made here to Eddington’s views, suggesting the connection with those early thoughts about physics and Mind. But I would now take Turing’s question more seriously, noting that the unpredictability of quantum mechanics lies in its still-mysterious ‘reduction’ process. The philosopher B.J. Copeland has also drawn attention to Turing’s 1951 sentence [Copeland, 1999], but in a context suggesting that Turing was connecting ‘randomness’ with ‘oracle-machines.’ This is unjustified: the point is that Turing was beginning to give active thought to the theory of wave-function reduction, as is described in Robin Gandy’s letter.

Another irony confronts us here: Robin Gandy had by that time switched entirely to becoming Turing’s successor in the field of mathematical logic where, in turn, Turing had abandoned an active interest. Hence the verbal explanations Turing gave to Robin, as mentioned in his letter, were probably mainly lost on him. Nevertheless Robin kept the postcards that Turing sent in 1954, and from which he quoted in his letter to Newman. (These survive in the King’s College archive, also being reproduced in [Hodges 1983, p. 513].) These are only scraps, but enough to suggest serious new directions. The language, depending on cryptic comments to be decoded by Robin in the light of their shared good humour, disguises their seriousness.

In [Hodges 1983, p. 495] I referred to Turing taking up Dirac’s theory of spinors; I have since learned from Sir Roger Penrose that Turing’s tensor analysis

notes [*General tensors in a group*, not dated, item C/13 in the Turing Archive, King's College, Cambridge] must be notes on Dirac's Cambridge lectures on quantum mechanics, which somehow he had found time to attend. The postcards also show Dirac's influence. One comment of Turing's, written as a sideline on one of the postcards, 'Does the gravitational constant decrease,' is indeed pure Dirac. But the other comments have greater originality.

The word 'fount' (i.e. a typographical 'font'), is probably Turing's own. It refers to the convention of using different fonts for tensor indices related to different symmetry groups. 'Particles are founts' implies a classification of particles by underlying symmetry groups, a key idea of elementary particle theory.

The statement about charge is essentially the idea of *gauge theory*, which in 1954 was taking on new life in the famous Yang–Mills paper of that year, which discussed non-Abelian gauge groups.

The reference to the 'interior of the light cone of the Creation' is not trivial. Light-ray-based descriptions were the key to the renaissance of General Relativity in 1954, a field in which little had happened in fifteen years but was soon transformed into a new arena both for differential geometry and astrophysics. (Clear evidence for the Big Bang was, however, eleven years away.)

The 'hyperboloids of wondrous light' are a mystery, but suggest some new geometrical theory for the propagation of quantum-mechanical wave functions.

Robin Gandy's letter then (in section 8) records Turing's dissatisfaction with standard 'Copenhagen' quantum mechanics, essentially because it gives no explanation of how or when the reduction of the wave-function is supposed to occur. The introduction of *non-linearity* (as mentioned in section 7) is consistent with how later thinkers have addressed this still-mysterious question.

To summarise: Turing was probably as much in touch with the problems of fundamental physics in 1954 as he was with those of mathematical logic in 1935, and the stage was set for a major creative act. Newman saw Turing's mind as still at 'the height of its power' in 1954. We can see these 'Messages from the Unseen World' as showing, or shadowing, what Turing might have done had he lived on.

But it is uncannily close to what Roger Penrose, just twenty-two at Turing's death, has since actually done. Turing's lines about 'boundary conditions' as opposed to 'differential equations' are clearly aimed at Eddington's mysticism in their reference to 'Religion.' Yet they can also be read seriously as a reference to the nature of physical law: physical science has so far rested on laws framed as differential equations but there is no absolute reason why this should remain the case. It is Penrose now who suggests that the union of gravity and quantum mechanics must involve some new kind of physical law, a boundary condition on space-time singularities which introduces asymmetry in time.

In [Hodges 1983, p. 514], I only hinted at how the physical programme sketched here by Turing has since been realised by Penrose. Since then Roger

Penrose has further suggested that an uncomputable element must enter into wave-function reduction in order to explain consciousness – we behold a mystery and close this Volume in natural wonder.

The letter written by Robin Gandy to Max Newman in June 1954

Dear Professor Newman

As perhaps you know I inherit Alan's books and manuscripts, and I should like to come and discuss him and his work and the possibilities of publication with you; I shall be coming to Wilmslow with Furbank on Saturday 26th June; would either the Sunday or Monday be a good day for me to come and see you? I am particularly anxious that the work on morphogenesis on a cylinder should not be lost; I gather from your obituary in the *Guardian* – which I much admired – that you feel the same. There are (or were) at least two draughts of it; but although I have looked at both of them, and had some of the theory expounded to me, I have not thought about the matter on my own; and I am hoping that there is someone who has a deeper knowledge than I who will be ready to prepare it for publication, or at least write a report on it. When I was staying with Alan the weekend before Whitsun he also told me more or less where the computations had got to; but since his methods were so individual, and he was unmethodical, I imagine it would be almost impossible for anyone to go on with the programme where he left off.

I take it that you will be writing his obituary for the F.R.S. and so I will offer you a miscellany of information; some of it you may know already.

1) Alan considered that his paper on ordinal logics had never received the attention it deserved (he wouldn't admit that it was a stinker to read). In particular:

a) There occurs in it, I believe for the first time, a scheme for the definition of numerical functions by transfinite recursion. The idea occurs in Hilbert's '*Über das Unendliche*' (1926); but the reference usually given is to Ackermann's '*Widerspruchsfreitbeweis*' (1940). Ackermann's scheme only differs from Alan's in that it uses particular primitive recursive orderings.

b) The paper shows clearly that statements about ordinals ('so and so can be proved by transfinite induction up to ordinal α ') depend very much on the particular formal representation of the ordinals which is used, and on the exact form of the rule of transfinite induction. Alan thought that this was always in danger of being forgotten by Goodstein, Kreisel, Schütte et al. My own belief is that the systems used at least by Kreisel, Schütte, Lorenz are perhaps sufficiently close for the required invariance to hold, and I hope one day to do some work on it. Alan's reaction was typically in the opposite direction – could one perhaps construct two different but rather similar looking systems which required different

ordinals for proving the same proposition. (This discussion arose out of Kreisel's criticism of my ω -consistency proof. Kreisel was at least right to the extent that it was more complicated than I had supposed, and that I now use a much larger ordinal).

2) Alan set some store by the definition of 'equivalence of two logical systems' which is given in his 'Practical Types'; and he intended the concealed type theory to be the beginning of a bridge between type theory and set theory; no one seems to have followed this up. I think that Labbe's work ('Transfinite types' JSL 1953) would be simpler if it had been done with Alan's nested types rather than Church's types.

3) I am going to write a book on the simple theory of types and some of his unpublished results will find their place in it. I think he really should have published his proof of the consistency of type theory with axiom of infinity but without extensionality, as it is a reasonably important result which does not seem to be widely known. (I think he did this work when at Princeton before the war; last time I asked him about it he looked for the manuscript but couldn't find it.)

4) I imagine you know that he produced an upper bound for the 'Skewes number' (is that the right spelling?) which was less than Skewes's but didn't like to cut in on him by publishing it.

5) I possess his working out of the strategy for 2-man I card poker, which he once talked about publishing. Of course it was put to a practical test and I seem to remember I was a few pence down and it was a few pence up after an evening's play (minimum stake 1/2 d (an old halfpenny – Ed.)).

6) I always hoped he would return one day to the practical problems of making a machine learn. There should be somewhere a copy of the report he wrote on this after his sabbatical year at Cambridge from the N.P.L.

7) During this spring he spent some time inventing a new quantum mechanics; it was not intended to be taken very seriously (almost in the 'for amusement only' class), although no doubt he hoped that something might turn up in it which could be taken seriously. But it did show him at his most lively and inventive; he said 'Quantum mechanists always seem to require infinitely many dimensions; I don't think I can cope with so many – I'm going to have about 100 or so – that ought to be enough don't you think?' Then he produced a slogan 'Description must be non-linear, prediction must be linear'. And then he bombarded me with a series of postcards. I have lost the first but here are the others:

No 2

Messages from the Unseen World

III The Universe is the interior of the light cone of the creation

IV Science is a differential Equation. Religion is a Boundary Condition.

(sgd) Arthur Stanley

(Ed. note: the postcards also have a side-comment: 'Does the gravitational constant decrease?' which is simply a reference to Dirac's theory of the time. A facsimile reproduction of the cards can be found in Hodges [1983, p. 513]).

No 3

V Hyperboloids of wondrous Light

Rolling for aye through Space and Time

Harbour those Waves which somehow might

Play out God's wondrous pantomime

(No 3 described rather more accurately than might appear his theory at that stage; was later chided for not having fully interpreted it.)

No 4

VI Particles are founts

VII Charge = e/π ang of character of a 2π rotation

VIII The Exclusion Principle is laid down purely for the benefit of the electrons themselves, who might be corrupted (and become dragons or demons) if allowed to associate too freely

8) A slightly more serious contribution to quantum mechanics uses 'the Turing Paradox'; it is easy to show using standard theory that if a system starts in an eigenstate of some observable, and measurements are made of that observable N times a second, then, even if the state is not a stationary one, the probability that the system will be in the same state after, say, 1 second, tends to one as N tends to infinity; i.e. that continual observation will prevent motion. Alan and I tackled one or two theoretical physicists with this, and they rather pooh-poohed it by saying that continual observation is not possible. But there is nothing in the standard books (e.g., Dirac's) to this effect, so that at least the paradox shows up an inadequacy of Quantum Theory as usually presented.

I thought you hit the nail on the head in the Guardian; the mark of his particular genius was that however abstract the topic he always had absolutely concrete examples and cases in mind; and this, of course, was why he found a lot of contemporary mathematics unsympathetic – he didn't like developing abstract concepts merely for their own sake.

Yours very sincerely

Robin Gandy



By courtesy of the National Portrait Gallery, London

ALAN MATHISON TURING

1912-1954

THE sudden death of ALAN TURING on 7 June 1954 deprived mathematics and science of a great original mind at the height of its power. After some years of scientific indecision, since the end of the war, Turing had found, in his chemical theory of growth and form, a theme that gave the fullest scope for his rare combination of abilities, as a mathematical analyst with a flair for machine computing, and a natural philosopher full of bold original ideas. The preliminary report of 1952, and the account that will appear posthumously, describe only his first rough sketch of this theory, and the unfulfilled design must remain a painful reminder of the loss that his early death has caused to science.

Alan Mathison Turing was born in London on 23 June 1912, the son of Julius Mathison Turing, of the Indian Civil Service, and of Ethel Sara Turing (*née* Stoney). The name 'Turing' is of Scottish, perhaps ultimately of Norman origin, the final *g* being an addition made by Sir William Turing, of Aberdeenshire, in the reign of James VI and I. The Stonies, an English-Irish family of Yorkshire origin, produced some distinguished physicists and engineers in the nineteenth century, three of whom became Fellows of the Society; and Edith A. Stoney was one of the early women equal-to-wranglers at Cambridge (bracketed with 17th Wrangler, 1893).

Alan Turing's interest in science began early and never wavered. Both at his preparatory schools and later at Sherborne, which he entered in 1926, the contrast between his absorbed interest in science and mathematics, and his indifference to Latin and 'English subjects' perplexed and distressed his teachers, bent on giving him a well-balanced education. Many of the characteristics that were strongly marked in his later life can already be clearly seen in remembered incidents of this time: his particular delight in problems, large or small, that enabled him to combine theory with the kind of experiments he could carry out with his own hands, with the help of whatever apparatus was at hand; his strong preference for working everything out from first principles instead of borrowing from others—a habit which gave freshness and independence to his work, but also undoubtedly slowed him down, and later on made him a difficult author to read. At school home-made experiments in his study did not fit well into the routine of the house: a letter from his housemaster mentions 'Heaven knows what witches' brew blazing on a naked wooden window sill'. But before he left school his abilities,

and his obvious seriousness of purpose, had won him respect and affection, and even tolerance for his own peculiar methods.

In 1931 he entered King's College, Cambridge, as a mathematical scholar. A second class in Part I of the Tripos showed him still determined not to spend time on subjects that did not interest him. In Part II he was a Wrangler, with 'b*', and he won a Smith's Prize in 1936. He was elected a Fellow of King's in 1935, for a dissertation on the Central Limit Theorem of probability (which he discovered anew, in ignorance of recent previous work).

It was in 1935 that he first began to work in mathematical logic, and almost immediately started on the investigation that was to lead to his best known results, on computable numbers and the 'Turing machine'. The paper attracted attention as soon as it appeared and the resulting correspondence led to his spending the next two years (1936-8) in Princeton, working with Professor Alonzo Church, the second of them as Procter Fellow.

In 1938 Turing returned to Cambridge; in 1939 the war broke out. For the next six years he was fully occupied with his duties for the Foreign Office. These years were happy enough, perhaps the happiest of his life, with full scope for his inventiveness, a mild routine to shape the day, and a congenial set of fellow-workers. But the loss to his scientific work of the years between the ages of 27 and 33 was a cruel one. Three remarkable papers written just before the war, on three diverse mathematical subjects, show the quality of the work that might have been produced if he had settled down to work on some big problem at that critical time. For his work for the Foreign Office he was awarded the O.B.E.

At the end of the war many circumstances combined to turn his attention to the new automatic computing machines. They were in principle realizations of the 'universal machine' which he had described in the 1937 paper for the purpose of a logical argument, though their designers did not yet know of Turing's work. Besides this theoretical link, there was a strong attraction in the many-sided nature of the work, ranging from electric circuit design to the entirely new field of organizing mathematical problems for a machine. He decided to decline an offer of a Cambridge University Lectureship, and join the group that was being formed at the National Physical Laboratory for the design, construction and use of a large automatic computing machine. In the three years (1945-8) that this association lasted he made the first plan of the ACE, the N.P.L.'s automatic computer, and did a great deal of pioneering work in the design of sub-routines.

In 1948 he was appointed to a Readership in the University of Manchester, where work was beginning on the construction of a computing machine by F. C. Williams and T. Kilburn. The expectation was that Turing would lead the mathematical side of the work, and for a few years he continued to work, first on the design of the sub-routines out of which the larger programmes for such a machine are built, and then, as this kind of work became standardized, on more general problems of numerical analysis. From 1950 onward he

turned back for a while to mathematics and finally to his biological theory. But he remained in close contact with the Computing Machine Laboratory, whose members found him ready to tackle the mathematical problems that arose in their work, and what is more, to find the answers, by that combination of powerful mathematical analysis and intuitive short cuts that showed him at heart more of an applied than a pure mathematician.

He was elected to the Fellowship of the Society in 1951.

For recreation he turned mostly to those 'home-made' projects and experiments, self-contained both in theory and practice, that have already been mentioned: they remained a ruling passion up to the last hours of his life. The rule of the game was that everything was to be done with the materials at hand, and worked out from data to be found in the house, or in his own head. This sort of self-sufficiency stood him in good stead in starting on his theory of 'morphogenesis', where the preliminary reading would have drowned out a more orthodox approach. In everyday life it led to a certain fondness for the gimcrack, for example the famous Bletchley bicycle, the chain of which would stay on if the rider counted his pedal-strokes and executed a certain manoeuvre after every seventeen strokes.

After the war, feeling in need of violent exercise, he took to long distance running, and found that he was very successful at it. He won the 3 miles and 10 miles championships of his club (the Walton Athletic Club), both in record time, and was placed fifth in the A.A.A. Marathon race in 1947. He thought it quite natural to put this accomplishment to practical use from time to time, for example by running some nine miles from Teddington to a technical conference at the Post Office Research Station in North London, when the public transport proved tedious.

In conversation he had a gift for comical but brilliantly apt analogies, which found its full scope in the discussions on 'brains *v.* machines' of the late 1940's. He delighted in confounding those who, as he thought, too easily assumed that the two things are separated by an impassable gulf, by challenging them to produce an examination paper that could be passed by a man, but not by a machine. The unexpected element in human behaviour he proposed, half seriously, to imitate by a random element, or roulette-wheel, in the machine. This, he said, would enable proud owners to say 'My machine' (instead of 'My little boy') 'said such a funny thing this morning'.

Those who knew Turing will remember most vividly the enthusiasm and excitement with which he would pursue any idea that caught his interest, from a conversational hare to a difficult scientific problem. Nor was it only the pleasure of the chase that inspired him. He would take the greatest pains over services, large or small, to his friends. His colleagues in the computing machine laboratory found him still as ready as ever with his help for their problems when his own interests were fully engaged with his bio-chemical theory; and, as another instance, he gave an immense amount of thought and care to the selection of the presents which he gave to his friends and their children at Christmas.

His death, at a time when he was fully absorbed in his scientific work, was a great and sad loss to his friends, as well as to the wider world of science.

Scientific work

The varied titles of Turing's published work disguise its unity of purpose. The central problem with which he 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 and 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.

1. Mathematical logic

The Hilbert decision-programme of the 1920's and 30's had for its objective the discovery of a general process, applicable to any mathematical theorem expressed in fully symbolical form, for deciding the truth or falsehood of the theorem. A first blow was dealt at the prospects of finding this new philosopher's stone by Gödel's incompleteness theorem (1931), which made it clear that truth or falsehood of A could not be equated to provability of A or not- A in any finitely based logic, chosen once for all; but there still remained in principle the possibility of finding a mechanical process for deciding whether A , or not- A , or neither, was formally provable in a given system. Many were convinced that no such process was possible, but Turing set out to demonstrate the impossibility rigorously. The first step was evidently to give a definition of 'decision process' sufficiently exact to form the basis of a mathematical proof of impossibility. To the question 'What is a "mechanical" process?' Turing returned the characteristic answer 'Something that can be done by a machine', and he embarked on the highly congenial task of analyzing the general notion of a computing machine. It is difficult to-day to realize how bold an innovation it was to introduce talk about paper tapes and patterns punched in them, into discussions of the foundations of mathematics. It is worth while quoting from his paper (1937a) the paragraph in which the computing machine is first introduced, both for the sake of its content and to give the flavour of Turing's writings.

'1. Computing machines'

'We have said that the computable numbers are those whose decimals are calculable by finite means. This requires rather more explicit definition. No real attempt will be made to justify the definitions given until we reach §9. For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited.'

'We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions q_1, q_2, \dots, q_k which will be called " m -configurations". The machine is supplied with a "tape" (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the r -th, bearing the symbol $\mathfrak{S}(r)$ which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its m -configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the m -configuration q_n and the scanned symbol $\mathfrak{S}(r)$. This pair $q_n, \mathfrak{S}(r)$ will be called the "configuration": thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (*i.e.* bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the m -configuration may be changed. Some of the symbols written down will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure.'

'It is my contention that these operations include all those which are used in the computation of a number. The defence of this contention will be easier when the theory of the machines is familiar to the reader.'

In succeeding paragraphs he gave arguments for believing that a machine of this kind could be made to do any piece of work which could be done by a human computer obeying explicit instructions given to him before the work starts. A machine of the kind he had described could be made for computing the successive digits of π , another for computing the successive prime numbers, and so forth. Such a machine is completely specified by a table, which states how it moves from each of the finite sets of possible 'configurations' to another. In the computations mentioned above, of π and of the successive primes, the machine may be supposed to be designed for its special purpose. It is supplied with a blank tape and started off. But we may also imagine a machine supplied with a tape already bearing a pattern which will influence its subsequent behaviour, and this pattern might be the table, suitably encoded, of a particular computing machine, X . It could be arranged that this tape would cause the machine, M , into which it was inserted to behave like machine X . Turing proved the fundamental result that there is a 'universal' machine, U (of which he gave the table), which can be made to do the

work of any assigned special-purpose machine, that is to say to carry out any piece of computing, if a tape bearing suitable 'instructions' is inserted into it. The machine U is so constructed that, presented with a tape bearing any arbitrary pattern it will move through a determinate, in general endless, succession of configurations; and it may or may not print at least one digit, 0 or 1. If it does, the pattern is 'circle-free'. It is therefore a problem, for which a decision process might be sought, to determine from inspection of a tape, whether or not it is circle-free. By means of a Cantor diagonal argument, Turing showed that no instruction-tape will cause the machine U to solve this problem, i.e. no pattern P is such that U , when presented with P followed by an arbitrary pattern γ , will print 0 if γ is 'circle-free', and 1 if it is not. If Turing's thesis is accepted, that the existence of a *method* for solving such a problem means the existence of a machine (or an instruction-tape for the universal machine U) that will solve it, it follows that the discovery of a process for discriminating between circle-free and other tapes is an insoluble problem, in an absolute and inescapable sense. From this basic insoluble problem it was not difficult to infer that the Hilbert programme of finding a decision method for the axiomatic system, \mathcal{Z} , of elementary number-theory, is also impossible.

In the application he had principally in mind, namely, the breaking down of the Hilbert programme, Turing was unluckily anticipated by a few months by Church, who proved the same result by means of his ' λ -calculus'. An offprint arrived in Cambridge just as Turing was ready to send off his manuscript. But it was soon realized that Turing's 'machine' had a significance going far beyond this particular application. It was shown by Turing (1937b) and others, that the definitions of 'general recursive' (by Gödel in 1931 and Kleene in 1935), ' λ -definable' (by Church in 1936) and 'computable' (Turing 1937a) have exactly the same scope, a fact which greatly strengthened the belief that they describe a fundamentally important body of functions. Turing's treatment has the merit of making a particularly convincing case for the acceptance of these and no other processes, as genuinely constructive; and it turned out to be well adapted for use in finding other insoluble problems, e.g. in the theory of groups and semi-groups.

Turing's other major contribution to this part of mathematical logic, the paper (1939) on systems of logic based on ordinals, has received less attention than (1937a), perhaps owing to its difficulty. The method of Gödel for constructing an undecidable sentence in any finitely based logic, L , i.e. a sentence expressible, but neither provable nor disprovable, in L , has led to the consideration of infinite families of 'logics', L_α , one for each ordinal α , where $L_{\alpha+1}$ is formed from L_α by the adjunction as an axiom of a sentence undecidable for L_α , if such exist, and L_α for limit ordinals α has as 'provable formulae', the union of the sets P_β ($\beta < \alpha$), where P_β is the set of provable formulae in L_β . The process must terminate for some $\gamma < \omega_1$, since the total set of formulae (which does not change) is countable. This procedure opens up the possibility of finding a logic that is complete, without violating Gödel's principle, since

L_α may not be finitely based if α is a limit ordinal. Rosser investigated this possibility in 1937, using the 'classical' non-constructive theory of ordinals. Turing took up the proposal, but with the proviso that, although some non-constructive steps must be made if a complete logic is to be attained, a strict watch should be kept on them. He first introduced a new theory of constructive ordinals, or rather of formulae (of Church's λ -calculus) representing ordinals; and he showed that the problem of deciding whether a formula represents an ordinal (in a plausible sense) is insoluble, in the sense of his earlier paper. A formula \mathbf{L} of the λ -calculus is a *logic* if it gives a means of establishing the truth of number-theoretic theorems; formally, if $\mathbf{L}(\mathbf{A})$ conv. 2 implies that $\mathbf{A}(\mathbf{n})$ conv. 2 for each \mathbf{n} representing a natural number. The *extent* of \mathbf{L} is the set of \mathbf{A} 's such that $\mathbf{L}(\mathbf{A})$ conv. 2, i.e. roughly speaking, the set of \mathbf{A} 's for which \mathbf{L} proves $\mathbf{A}(\mathbf{n})$ true for all \mathbf{n} . An *ordinal logic* is now defined to be a formula Λ , such that $\Lambda(\Omega)$ is a logic whenever Ω represents an ordinal; and Λ is *complete* if every number-theoretic theorem that is true, is probable in $\Lambda(\Omega)$ for some Ω , i.e. if given \mathbf{A} such that $\mathbf{A}(\mathbf{n})$ conv. 2 for each \mathbf{n} representing a natural number, $\Lambda(\Omega, \mathbf{A})$ conv. 2 for some Ω (depending on \mathbf{A}). It is next shown, by an example, that formulae Ω_1, Ω_2 may represent the same ordinal, but yet make $\Lambda(\Omega_1)$ and $\Lambda(\Omega_2)$ different logics, in the sense that they have different extents. An ordinal logic for which this cannot happen is *invariant*. It is only in invariant logics that the 'depth' of a theorem can be measured by the size of the ordinal required for its proof. The main theorems of the paper state (1) that complete ordinal logics and invariant ordinal logics exist, (2) that no complete and invariant ordinal logic exists.

This paper is full of interesting suggestions and ideas. In §4 Turing considers, as a system with the minimal departure from constructiveness, one in which number-theoretic problems of some class are arbitrarily assumed to be soluble: as he puts it, 'Let us suppose that we are supplied with some unspecified means of solving number-theoretic problems; a kind of oracle, as it were.' The availability of the oracle is the 'infinite' ingredient necessary to escape the Gödel principle. It also obviously resembles the stages in the construction of a proof by a mathematician where he 'has an idea', as distinct from making mechanical use of a method. The discussion of this and related matters in §11 ('The purpose of ordinal logics') throws much light on Turing's views on the place of intuition in mathematical proof. In the final rather difficult §12 the idea adumbrated by Hilbert in 1922 of recursive definitions of order-types other than ω received its first detailed exposition.

Besides these two pioneering works, and the papers (1937b, c), arising directly out of them, Turing published four papers of predominantly logical interest. (A) The paper (1942a), with M. H. A. Newman, on a formal question in Church's theory of types. (B) A 'practical form of type-theory' (1948b) is intended to give Russell's theory of types a form that could be used in ordinary mathematics. Since the more flexible Zermelo-von Neumann set-theory has been generally preferred to type-theory by mathematicians, this paper has received little attention. It contains a number of interesting ideas,

in particular a definition of 'equivalence' between logical systems (p. 89). (C) The use of dots as brackets (1942b), an elaborate discussion of punctuation in symbolic logic. Finally, (D) contains the proof (1950a) of the insolubility of the word-problem for semi-groups with cancellation. A finitely generated semi-group *without* cancellation is determined by choosing a finite set of pairs of words, (A_i, B_i) ($i=1, \dots, k$) of some alphabet, and declaring two words to be 'equivalent' if they can be proved so by the use of the equations $PA_iQ = PB_iQ$, where P and Q can be arbitrary words (possibly empty). The word-problem for such a semi-group is to find a process which will decide whether or not two given words are equivalent. The insolubility of this problem can be brought into relation with the fundamental insoluble machine-tape problem. The table of a computing machine states, for each configuration, what is the configuration that follows it. Since a configuration can be denoted by a 'word', in letters representing the internal configurations and tape-symbols, this table gives a set of pairs of words which, when suitably modified, determine a semi-group with insoluble word problem. So much was proved by E. L. Post in 1947. The question becomes much more difficult if the semi-group is required to satisfy the cancellation laws, ' $AC = BC$ implies $A = B$ ' and ' $CA = CB$ implies $A = B$ ', since now a condition is imposed on account of its mathematical interest, and not because it arises naturally from the machine interpretation. This was the step taken by Turing in (1950a). (For a helpful discussion and analysis of this difficult paper see the long review by W. W. Boone, *J. Symbolic Logic*, **17** (1952) 74.)

2. Three mathematical papers

Shortly before the war Turing made his only contributions to mathematics proper.

The paper 1938a contains an interesting theorem on the approximation of Lie groups by finite groups: if a (connected) Lie group, L , can for arbitrary $\varepsilon > 0$ be ε -approximated by a finite group whose multiplication law is an ε -approximation to that of L , in the sense that the two products of any two elements are within ε of each other, then L must be both compact and abelian. The theory of representations of topological groups is used to apply Jordan's theorem on the abelian invariant subgroups of finite groups of linear transformations.

Paper (1938b) lies in the domain of classical group theory. Results of R. Baer on the extensions of a group are re-proved by a more unified and simpler method.

Paper (1943)—submitted in 1939, but delayed four years by war-time difficulties—shows that Turing's interest in practical computing goes back at least to this time. A method is given for the calculation of the Riemann zeta-function, suitable for values of t in a range not well covered by the previous work of Siegel and Titchmarsh. The paper is a straightforward but highly skilled piece of classical analysis. (The post-war paper (1953a) describes an

attempt to apply a modified form of this process, which failed owing to machine trouble.)

3. *Computing machines*

Apart from the practical 'programmer's handbook', only two published papers (1948a and 1950b) resulted from Turing's work on machines. When binary fractions of fixed length are used (as they must be on a computing machine) for calculations involving a very large number of multiplications and divisions, the necessary rounding-off of exact products introduces cumulative errors, which gradually consume the trustworthy digits as the computation proceeds. The paper (1948a) investigates questions of the following type: how many figures of the answer are trustworthy if k figures are retained in solving n linear equations in n unknowns? The answer depends on the method of solution, and a number of different ones are considered. In particular it is shown that the ordinary method of successive elimination of the variables does not lead to the very large errors that had been predicted, save in exceptional cases which can be specified.

The other paper (1950b) arising out of his interest in computing machines is of a very different nature. This paper, on computing machines and intelligence, contains Turing's views on some questions about which he had thought a great deal. Here he elaborates his notion of an 'examination' to test machines against men, and he examines systematically a series of arguments commonly put forward against the view that machines might be said to think. Since the paper is easily accessible and highly readable, it would be pointless to summarize it. The conversational style allows the natural clarity of Turing's thought to prevail, and the paper is a masterpiece of clear and vivid exposition.

The proposals (1953b) for making a computing machine play chess are amusing, and did in fact produce a defence lasting 30 moves when the method was tried against a weak player; but it is possible that Turing underestimated the gap that separates combinatory from position play.

4. *Chemical theory of morphogenesis*

For the following account of Turing's final work I am indebted to Dr N. E. Hoskin, who with Dr B. Richards is preparing an edition of the material for publication.

The work falls into two parts. In the first part, published (1952) in his lifetime, he set out to show that the phenomena of morphogenesis (growth and form of living things) could be explained by consideration of a system of chemical substances whose concentrations varied only by means of chemical reactions, and by diffusion through the containing medium. If these substances are considered as form-producers (or 'morphogens' as Turing called them) they may be adequate to determine the formation and growth of an organism, if they result in localized accumulations of form-producing substances. According to Turing the laws of physical chemistry are sufficient to

account for many of the facts of morphogenesis (a view similar to that expressed by D'Arcy Thompson in *Growth and form*).

Turing arrived at differential equations of the form

$$\frac{\partial X_i}{\partial t} = f_i(X_1, \dots, X_n) + \mu \nabla^2 X_i, \quad (i = 1, \dots, n)$$

for n different morphogens in continuous tissue; where f_i is the reaction function giving the rate of growth of X_i , and $\nabla^2 X_i$ is the rate of diffusion of X_i . He also considered the corresponding equations for a set of discrete cells. The function f_i involves the concentrations, and in his 1952 paper Turing considered the X_i 's as variations from a homogeneous equilibrium. If, then, there are only small departures from equilibrium, it is permissible to linearize the f_i 's, and so linearize the differential equations. In this way he was able to arrive at the conditions governing the onset of instability. Assuming initially a state of homogeneous equilibrium disturbed by random disturbances at $t = 0$, he discussed the various forms instability could take, on a continuous ring of tissue. Of the forms discussed the most important was that which eventually reached a pattern of stationary waves. The botanical situation corresponding to this would be an accumulation of the relevant morphogen in several evenly distributed regions around the ring, and would result in the main growth taking place at these points. (The examples cited are the tentacles of *Hydra*, and whorled leaves.) He also tested the theory by obtaining numerical solutions of the equations, using the electronic computer at Manchester. In the numerical example, in which two morphogens were supposed to be present in a ring of twenty cells, he found that a three or four lobed pattern would result. In other examples he found two-dimensional patterns, suggestive of dappling; and a system on a sphere gave results indicative of gastrulation. He also suggested that stationary waves in two dimensions could account for the phenomena of phyllotaxis.

In his later work (as yet unpublished) he considered quadratic terms in the reaction functions in order to take account of larger departures from the state of homogeneous equilibrium. He was attempting to solve the equations in two dimensions on the computer at the time of his death. The work is in existence, but unfortunately is in a form that makes it extremely difficult to discover the results he obtained. However, B. Richards, using the same equations, investigated the problem in the case where the organism forms a spherical shell and also obtained numerical results on the computer. These were compared with the structure of *Radiolaria*, which have spikes on a basic spherical shell, and the agreement was strikingly good. The rest of this part of Turing's work is incomplete, and little else can be obtained from it. However, from Richards's results it seems that consideration of quadratic terms is sufficient to determine practical solutions, whereas linear terms are really only sufficient to discuss the onset of instability.

The second part of the work is a mathematical discussion of the geometry of phyllotaxis (i.e. of mature botanical structures). Turing discussed many

ways of classifying phyllotaxis patterns and suggested various parameters by which a phyllotactic lattice may be described. In particular, he showed that if a phyllotactic system is Fibonacci in character, it will change, if at all, to a system which has also Fibonacci character. This is in accordance with observation. However, most of this section was intended merely as a description preparatory to his morphogenetic theory, to account for the facts of phyllotaxis; and it is clear that Turing did not intend it to stand alone.

The wide range of Turing's work and interests have made the writer of this notice more than ordinarily dependent on the help of others. Among many who have given valuable information I wish to thank particularly Mr R. Gandy, Mr J. H. Wilkinson, Dr B. Richards and Dr N. E. Hoskin; and Mrs Turing, Alan Turing's mother, for constant help with biographical material.

M. H. A. NEWMAN

BIBLIOGRAPHY

- 1935a. On the Gaussian Error Function (King's College Fellowship Dissertation).
- 1935b. Equivalence of left and right almost periodicity. *J. Lond. Math. Soc.* **10**, 284.
- 1937a. On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* (2), **42**, 230.
- 1937b. Computability and λ -definability. *J. Symbolic logic*, **2**, 153.
- 1937c. The p -function in λ - K -conversion. *J. Symbolic logic*, **2**, 164.
- 1937d. Correction to 1937a. *Proc. Lond. Math. Soc.* (2), **43**, 544.
- 1938a. Finite approximations to Lie groups. *Ann. Math.*, Princeton, **39**, 105.
- 1938b. The extensions of a group. *Comp. Math.* **5**, 357.
1939. Systems of logic based on ordinals. *Proc. Lond. Math. Soc.* (2), **45**, 161.
- 1942a. (With M. H. A. NEWMAN.) A formal theorem in Church's theory of types. *J. Symbolic logic*, **7**, 28.
- 1942b. The use of dots as brackets in Church's system. *J. Symbolic logic*, **7**, 146.
- 1943*. A method for the calculation of the zeta-function. *Proc. Lond. Math. Soc.* (2), **48**, 180.
- 1948a. Rounding-off errors in matrix processes. *Quart. J. Mech. App. Math.* **1**, 287.
- 1948b. Practical forms of type-theory. *J. Symbolic logic*, **13**, 80.
- 1950a. The word problem in semi-groups with cancellation. *Ann. Math.*, Princeton, **52**, 491.
- 1950b. Computing machinery and intelligence. *Mind*, **59**, 433.
- [1950c]. *Programmers' Handbook for the Manchester electronic computer*.
1952. The chemical basis of morphogenesis. *Phil. Trans. B*, **237**, 37.
- 1953a. Some calculations of the Riemann zeta-function. *Proc. Lond. Math. Soc.* (3), **3**, 99.
- 1953b. Digital computers applied to games: chess, pp. 288-295 of *Faster than thought*, ed. B. V. Bowden, Pitman, London.
1954. Solvable and unsolvable problems. *Sci. News*, **31**, 7.
[A second paper on morphogenesis is being prepared for publication by N. E. Hoskin and B. Richards, based on work left by Turing.]

* Received four years earlier (7 March 1939).

This Page Intentionally Left Blank

BIBLIOGRAPHY

ACKERMANN, W.

1928 Zum Hilbertschen Aufbau der reellen Zahlen

Math. Annal. **99**, 118–133. Translation in VAN HEIJENOORT 1967.

ANDERSON, A.R. (Ed.)

1964 *Minds and Machines*

(Prentice-Hall, Inc, NJ)

BABBAGE, C.

1994 *Passages from the Life of a Philosopher*

(London). Selections reprinted in MORRISON and MORRISON 1961.

BARENDEGT, H.P.

1984 *The Lambda Calculus. Its Syntax and Semantics*, 2nd ed.

(North-Holland Publ. Co., Amsterdam)

BURKS, A. W.

1980 From ENIAC to the stored-program computer: Two revolutions in computers
In: *A History of Computing in the Twentieth Century*

(Academic Press, New York) 541–550.

CHURCH, A.

1936 An unsolvable problem of elementary number theory

Amer. J. Math., 345–363. (Reprinted in DAVIS 1965.)

1936a A note on the Entscheidungsproblem

J. Symb. Logic **1**, 40–41.

1940 A formulation of the simple theory of types

J. Symb. Logic **5**, 56–68.

CHURCH, A. AND S.C. KLEENE

1936 Formal definitions in the theory of ordinal numbers

Fund. Math. **2**, 11–21.

COPELAND, B.J. (ED.)

1999 A lecture and two radio broadcasts on machine intelligence by Alan Turing

In: F. Furukawa, D. Michie, S. Muggleton (Eds.), *Machine Intelligence* **15**
(Oxford University Press, Oxford)

CURRY, H.B.

- 1937 On the use of dots in logical expressions
J. Symb. Logic **2**, 26–28.

DAVIS, M.

- 1965 *The Undecidable*
(Raven Press, New York)

- 1982 Why Gödel didn't have Church's Thesis
Inf. Contr. **54**, 3–24.
1988 Mathematical Logic and the Origin of Modern Computers
In: *The Universal Turing Machine – A Half-Century Survey*, 141–174.
(HERKEN 1988 and 1994 below.)

DAVIS, M., YU. MATIJASEVIC AND J. ROBINSON

- 1976 Hilbert's tenth problem. Diophantine equations: Positive aspects of a negative solution
In: F.E. Browder (Ed.), *Mathematical Developments Arising from Hilbert Problems*
(AMS, Providence, RI) 323–378.

ECKERT, J.P.

- 1980 *The ENIAC. A History of Computing in the Twentieth Century*
(Academic Press, New York) 525–539.

FEFERMAN, S.

- 1962 Transfinite recursive progressions of axiomatic theories
J. Symb. Logic **27**, 259–316.
1962a Classifications of recursive functions by means of hierarchies
Trans. Amer. Math. Soc. **104**, 101–122.
1964 Systems of predicative analysis
J. Symb. Log. **29**, 1–30.
1986 The life and work of Kurt Gödel
See GÖDEL, 1–36.
1988 Turing in the land of $O(z)$
In: *The Universal Turing Machine – A Half-Century Survey*, 113–147.
(HERKEN 1988 and 1994 below.)
1991 Reflecting on incompleteness
J. Symb. Logic **56**, 1–49.
1996 Gödel's program for new axioms. Why, where, how and what?
In: P. Hájek (Ed.), *Gödel '96*, Lecture Notes in Logic **6**, 3–22.

GANDY, R.O.

- 1956 On the axiom of extensionality, Part I
J. Symb. Logic **21**, 36–48;
Part II, *J. Symb. Logic* **24** (1959), 287–300.
- 1981 Church's Thesis and principles for mechanisms
In: J. Barwise, J.J. Keisler and K. Kunen (Eds.), *The Kleene Symposium* (North-Holland, Amsterdam) 123–145.
- 1988 The confluence of ideas in 1936
In: *The Universal Turing Machine – A Half-Century Survey* (HERKEN 1988 and 1994 below.)
- 2001 Axioms of infinity in Church's type theory
In: *Logic, Meaning and Computation: Essays in Memory of Alonzo Church*, C. Anthony Anderson and M. Zeleny (Eds.) (Kluwer Academic Publishers, Dordrecht) pp. 141–149.

GÖDEL, K.

- 1986 *Collected Works, Volume 1. Publications 1929–1936*
Eds. S. Feferman, J.W. Dawson, Jr., S.C. Kleene, G.H. Moore, R.M. Solovay and J. van Heijenoort (Oxford University Press, New York, 1986)
- 1990 ditto, *Volume 2. Publications 1938–74*.
- 1995 ditto, *Volume 3. Publications 1938–74*.

HERKEN, R.

- 1988 *The Universal Turing Machine – A Half-Century Survey*, 1st edition (OUP, Oxford)
- 1994 *Ditto*. 2nd edition: Springer-Verlag, New York.
- 1998 *Ditto*. (Verlag Kammerer & Unverzagt, Berlin).

HILBERT, D.

- 1926 Über das Unendliche
Math. Annal. **102**, 1–9.

HILBERT, D. AND ACKERMANN, W.

- 1950 *Introduction to Mathematical Logic*
(New York)

HODGES, A.

- 1983 *Alan Turing: The Enigma*
(Burnett Books, London and Simon and Schuster, New York).
A number of other editions, in paperback and in different countries, have subsequently been published.

1997 *Turing. A Natural Philosopher*

In the Great Philosophers series, Eds. Ray Monk and Frederic Raphael
(Phoenix, London)

KLEENE, S.C.

1952 *Introduction to Metamathematics*

(Van Nostrand)

1981 Origins of recursive function theory

Ann. Hist. Comp. 3, 52–67.

KREISEL, G.

1958 Ordinal logics and the characterization of informal methods of proof

In: *Proceedings of the International Congress of Mathematics at Edinburgh*, 289–299.

1970 Principles of proof and ordinals implicit in given concepts

In: A. Kino, J. Myhill and R.E. Vesley (Eds.), *Intuitionism and Proof Theory*

(North-Holland, Amsterdam)

KRIPKE, S.A.

1982 *Wittgenstein on Rules and Private Languages*

(Blackwell, Oxford)

COUNTESS OF LOVELACE, AUGUSTA ADA

1843 Sketch of the Analytical Engine Invented by Charles Babbage, by
L.F. Menabrea of Turin, Officer of the Military Engineers, with notes upon
the Memoir by the Translator

Taylor's Scientific Memoirs 3.

Reprinted in MORRISON and MORRISON 1961.

MAUCHLY, J.W.

1980 *The ENIAC. A History of Computing in the Twentieth Century*

(Academic Press, New York) 541–550.

MENABRAE, L.F.

1842 Notions sur la machine analytique de M. Charles Babbage

Bibliothèque Universelle de Genève 82, 325–376.

Translated in LOVELACE 1843.

METROPOLIS, N., HOWLETT, J. AND ROTA, G.-C. (Eds.)

1984 *A History of Computing in the Twentieth Century*

(Academic Press, New York)

MORRISON, P. AND MORRISON, E. (Eds.)

- 1961 *Charles Babbage and his Calculating Engines: Selected Writings by Charles Babbage and Others*
(Dover, New York)

NEWMAN, M.H.A.

- 1943 Stratified systems of logic
Proc. Camb. Phil. Soc. **39**, 69–83.
1948 General principles of the design of all-purpose computing machines
Proc. Roy. Soc. (A) **195**, 271–274.
1985 Alan Mathison Turing 1912–1954
Bibliographical Memoirs of Fellows of the Royal Society **1** (1955) 252–263.

NEWMAN, M.H.A. AND TURING, A.M.

- 1942 A formal theorem in Church's theory of types
J. Symb. Logic **7**, 28–33.

PARIS, J.B. AND HARRINGTON, L.

- 1977 A mathematical incompleteness in Peano arithmetic
In: J. Barwise (Ed.), *Handbook of Mathematical Logic*
(North-Holland, Amsterdam) 1133–1142.

PENROSE, R.

- 1988 On the physics and mathematics of thought
In: *The Universal Turing Machine – A Half-Century Survey*
(HERKEN 1988 and 1994 above, pp. 491–522.)
1989 *The Emperor's New Mind*
(Oxford University Press)

POST, E.L.

- 1943 Formal reductions of the general combinatorial decision problem
Amer. J. Math. **65**, 197–215.
1944 Recursively enumerable sets of positive integers and their decision problems
Bull. Amer. Math. S. **50**, 248–316.
Reprinted in DAVIS 1965.

RANDELL, B.

- 1972 On Alan Turing and the origins of digital computers
In: B. Melzer and D. Michie (Eds.), *Mach. Intell.* **7**, Edinburgh University Press, 3–20.

- 1980 The COLOSSUS
In: METROPOLIS et al., 47–92.
- 1982 *The Origins of Digital Computers, Selected Papers*, B. Randell (Ed.)
3rd edition (Springer-Verlag, Berlin)

TURING, A.M.

- 1937 On computable numbers, with an application to the Entscheidungsproblem
Proc. Lond. Math. Soc. (2) **42** (1936–7), 230 – 265.
— A correction. *ibid.* : **43**, 544–546.
- 1937a Computability and λ -definability
J. Symb. Logic **2** (1937), 153–163.
The p -function in λ - K conversion
J. Symb. Logic **2**, 164.
- 1939 Systems of logic based on ordinals
Proc. Lond. Math. Soc. (2) **45**, 161–228.
- 1942 The use of dots as brackets in Church's system
J. Symb. Logic **7**, 146–156.
- 1942a Visit to National Cash Register Corporation of Dayton, Ohio
A report by Turing of December 1942
In 'Bombe Correspondence' (Crane Collection) CSNG LIB, Box 139, RG 38, Records of the Office of Naval Intelligence.
- 1945 Proposal for development in the mathematics division of an Automatic Computing Engine (ACE)
Report E882, Executive Committee NPL. Re-issued in 1972 with a foreword by D.W. Davies as NPL report, *Com. Sci.* **57**. Printed in TURING 1986.
- 1947 The automatic computing engine
Lecture given to the London Mathematical Society on February 20, 1947.
Printed in TURING 1986.
- 1948 Practical forms of type theory
J. Symb. Logic **13**, 80–94.
- 1950 Computing machinery and intelligence
Mind **59**, 433–460.
Reprinted in ANDERSON 1964.
- 1986 *A.M. Turing's ACE Report of 1946 and Other Papers*
R.E. Carpenter and R.W. Doran (Eds.)
(The MIT Press, Cambridge, MA; and Tomash Publishers, Los Angeles)

VON NEUMANN, J.

1927 *Zur Hilbertschen Beweistheorie*

Math. Z. **26**, 1–46.

Reprinted in VON NEUMANN 1961.

1945 First draft of a report on the EDVAC

Contract No. W-670-ORD-492, Moore School of Electrical Engineering,
University of Pennsylvania.

Excerpts printed in RANDELL 1982.

1951 The general and logical theory of automata

In: L.A. Jeffress (Ed.), *Cerebral Mechanisms in Behaviour, the Hixon Symposium September 1948, Pasadena* (Wiley & Sons, New York).

Reprinted in VON NEUMANN 1961, Vol. V.

1961 *Collected Works*, Vols. I–VI, Ed. A.H. Taub

(Pergamon Press, Oxford)

WITTGENSTEIN, L.

1976 *Wittgenstein's lectures on the Foundations of mathematics, Cambridge 1939*

Ed. C. Diamond

(Horrocks, Sussex; The Harvester Press)

1979 *Ludwig Wittgenstein and the Vienna Circle: Conversations recorded by Friedrich Waismann*

Ed. Brian McGuinness

(Basil Blackwell; Oxford)

This Page Intentionally Left Blank

CONTENTS OF OTHER VOLUMES

Mechanical Intelligence Edited by D.C. INCE

	Preface	vii
	Alan Mathison Turing – Chronology	viii
	Introduction	ix
1945	Proposals for Development in the Mathematics Division of an Automatic Computing Engine (ACE) Report to the Executive Committee of the National Physics Laboratory. In: B.E. Carpenter and R.N. Doran (Editors), A.M. Turing's ACE Report of 1946 and Other Papers (MIT Press, Cambridge, MA, 1986) Chapter 2, pp. 20–105	1
1947	Lecture to the London Mathematical Society on 20 February 1947 In: B.E. Carpenter and R.N. Doran (Editors), A.M. Turing's ACE Report of 1946 and Other Papers (MIT Press, Cambridge, MA, 1986) Chapter 3, pp. 106–124	87
1948	Intelligent Machinery Report, National Physics Laboratory. In: B. Meltzer and D. Michie (Editors), Machine Intelligence 5 (Edinburgh University Press, Edinburgh, 1969) pp. 3–23	107
1949	Checking a Large Routine Paper, EDSAC Inaugural Conference, 24 June 1949. In: Report of a Conference on High Speed Automatic Calculating Machines, pp. 67–69	129
1950	Computing Machinery and Intelligence MIND LIX, pp. 433–460	133
1953	Digital Computers Applied to Games In: B.V. Bowden (Editor), Faster Than Thought (Pitman, London, 1953) Chapter 25, pp. 286–310	161
1954	Solvable and Unsolvable Problems Science News 31, pp. 7–23	187
	Notes	205
	Bibliography	217
	Index	223

Pure Mathematics
Edited by J.L. BRITTON

Preface	vii	
Alan Mathison Turing – Chronology	viii	
Introduction	ix	
Postscript	xvii	
Remarks on Turing's Dissertation	xix	
Turing's Published Papers		
1935	Equivalence of Left and Right Almost Periodicity J. London Math. Soc. 10, pp. 284–285	1
1938 A	Finite Approximations to Lie Groups Ann. of Math. 39 (1), pp. 105–111	3
1938 B	The Extensions of a Group Compositio Math. 5, pp. 357–367	11
1943	A Method for the Calculation of the Zeta-function Proc. London Math. Soc. (2) 48, pp. 180–197	23
1948	Rounding-off Errors in Matrix Processes Quart. J. Mech. Appl. Math. 1, pp. 287–308	41
1950	The Word Problem in Semi-groups With Cancellation Ann. of Math. 52 (2), pp. 491–505	63
1953	Some Calculations of the Riemann Zeta-function Proc. London Math. Soc. (3) 3, pp. 99–117	79
1954	Solvable and Unsolvable Problems Science News 31, pp. 7–23	99
Unpublished Papers		
I	A Note on Normal Numbers	117
II	The Word Problem in Compact Groups	121
III	On Permutation Groups	125
IV	The Difference $\psi(x) - x$	147
With SKEWES, S.		
V	On a Theorem of Littlewood	153
Related Papers		
BOONE, W.W.		
1958	An Analysis of Turing's "The Word Problem in Semigroups with Cancellation" Ann. of Math. 67 (1), pp. 195–202	175

COHEN, A.M. and MAYHEW, M.J.E.		
1968	On the Difference $\pi(x) - \text{li } x$	183
	Proc. London Math. Soc. (3) 18, pp. 691–713	

Turing's Statistical Work

GOOD, I.J.		
1979	Studies in the History of Probability and Statistics.	
	XXXVII. A.M. Turing's Statistical Work in World War II	207
	Biometrika 66 (2), pp. 393–396	
1989	Introductory Remarks for the Article in Biometrika 66 (1979)	211
	(Specially written for this volume)	
	Notes and Summaries	225
	Bibliography	275
	Index	283

Morphogenesis Edited by P.T. SAUNDERS

Preface	vii
Alan Mathison Turing – Chronology	viii
Preface to this volume	ix
Introduction	xi
1952	
The Chemical Basis of Morphogenesis	
Phil. Trans. R. Soc. London B 237, pp. 37–72	1
A Diffusion Reaction Theory of Morphogenesis in Plants (with C.W. Wardlaw)	37
Morphogen Theory of Phyllotaxis	49
I. Geometrical and Descriptive Phyllotaxis	49
II. Chemical Theory of Morphogenesis	88
III. A Solution of the Morphogenetical Equations for the Case of Spherical Symmetry (with B. Richards)	107
Outline of the Development of the Daisy	119
Bibliography	125
Index	129

This Page Intentionally Left Blank

APPENDIX: MATTERS ARISING FROM EARLIER VOLUMES

This section lists some other items, which were not referred to in *Collected Works of A.M. Turing: Machine Intelligence* (1992), but which we feel deserve a mention here for the sake of completeness; that volume will be referred to as *MI*.

1. *The Automatic Computing Engine*, Lectures given at the Ministry of Supply, December 1946 and January 1947, by Turing and J.H. Wilkinson. A report on these was written by T.H. Marshall, Military College of Science, Shrivenham, February 1947. This is of only marginal importance because the lectures were about the technical details of the ACE hardware design, and only two of the lectures were by Turing. Turing's full report was published in *MI*, p. 1–86.

2. *Intelligent Machinery*, the report written by Turing for the NPL at the end of summer 1948. There is a typescript in the Kings College Archive. This paper was first published in 1968, within the book *Cybernetics: Key Papers*, eds. C.R. Evans and A.D.J. Robertson, University Park Press, Baltimore, Maryland and Manchester (1968), but this original appears to have moved into obscurity.

The paper was again published in *Machine Intelligence 5*, p. 3–23, Edinburgh University Press (1969), with an introduction by Donald Michie. This commonly known version was then understandably reproduced in *MI*, p. 107–127. Unfortunately, this 1969 edition has some misprints, in particular it gets the diagrams wrong, so it is worth knowing about the original edition.

3. *Intelligent Machinery, A Heretical Theory*, a talk given by Turing at Manchester. There is a typescript in the Kings College Archive and it was included in the Memoir *A.M. Turing*, written by his mother, Sara Turing, and published by Heffer & Sons: Cambridge 1959 (p.128–134).

4. Radio Broadcasts in 1951 and 1952. Part of a series on *Automatic Calculating Machines* involving other participants, Turing's first lecture on May 15th, 1951, bore the subtitle *Can Digital Computers Think?* A four-cornered discussion of the same topic followed on January 14th, 1952.

5. Turing wrote a subsection of *Chapter 25, Digital Computers applied to Games* in *Faster Than Thought*, ed. B.V. Bowden, Pitman: London (1953). In *MI* the whole of the chapter was ascribed to Turing, but in fact the section on *Draughts* was written by Christopher Strachey, that on *NIM* by Audrey Bates and the introductory remarks presumably by the editor.

This Page Intentionally Left Blank