Proofs and Computations

Driven by the question "What is the computational content of a (formal) proof?", this book studies fundamental interactions between proof theory and computability. It provides a unique self-contained text for advanced students and researchers in mathematical logic and computer science.

Part 1 covers basic proof theory, computability and Gödel's theorems. Part 2 studies and classifies provable recursion in classical systems, from fragments of Peano arithmetic up to Π^1_1-CA_0. Ordinal analysis and the (Schwichtenberg–Wainer) subrecursive hierarchies play a central role, and are used in proving the "modified finite Ramsey" and "extended Kruskal" independence results for PA and Π^1_1-CA_0. Part 3 develops the theoretical underpinnings of the first author's proof-assistant MINLOG. Three chapters cover higher-type computability via information systems, a constructive theory TCF of computable functionals, realizability, Dialectica interpretation, computationally significant quantifiers and connectives, and polytime complexity in a two-sorted, higher-type arithmetic with linear logic.

HELMUT SCHWICHTENBERG is an Emeritus Professor of Mathematics at Ludwig-Maximilians-Universität Munchen. He has recently developed the "proof-assistant" MINLOG, a computer-implemented logic system for proof/program development and extraction of computational content.

STANLEY S. WAINER is an Emeritus Professor of Mathematics at the University of Leeds and a past-President of the British Logic Colloquium.

PERSPECTIVES IN LOGIC

The *Perspectives in Logic* series publishes substantial, high-quality books whose central theme lies in any area or aspect of logic. Books that present new material not now available in book form are particularly welcome. The series ranges from introductory texts suitable for beginning graduate courses to specialized monographs at the frontiers of research. Each book offers an illuminating perspective for its intended audience.

The series has its origins in the old *Perspectives in Mathematical Logic* series edited by the Ω-Group for "Mathematische Logik" of the Heidelberger Akademie der Wissenschaften, whose beginnings date back to the 1960s. The Association for Symbolic Logic has assumed editorial responsibility for the series and changed its name to reflect its interest in books that span the full range of disciplines in which logic plays an important role.

For more information, see www.aslonline.org/books_perspectives.html

PERSPECTIVES IN LOGIC

Proofs and Computations

HELMUT SCHWICHTENBERG
Ludwig-Maximilians-Universität Munchen

STANLEY S. WAINER
University of Leeds

ASSOCIATION FOR SYMBOLIC LOGIC

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town,
Singapore, São Paulo, Delhi, Mexico City

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521517690

First published 2012

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

ISBN 978-0-521-51769-0 Hardback

To Ursula and Lib
for their love and patience

In memory of our teachers
Dieter Rödding (1937–1984)
Martin H. Löb (1921–2006)

CONTENTS

Part 2. Provable recursion in classical systems

Part 3. Constructive logic and complexity

PREFACE

This book is about the deep connections between proof theory and recursive function theory. Their interplay has continuously underpinned and motivated the more constructively orientated developments in mathematical logic ever since the pioneering days of Hilbert, Gödel, Church, Turing, Kleene, Ackermann, Gentzen, Péter, Herbrand, Skolem, Malcev, Kolmogorov and others in the 1930s. They were all concerned in one way or another with the links between logic and computability. Gödel's theorem utilized the logical representability of recursive functions in number theory; Herbrand's theorem extracted explicit loop-free programs (sets of witnessing terms) from existential proofs in logic; Ackermann and Gentzen analysed the computational content of ε-reduction and cut-elimination in terms of transfinite recursion; Turing not only devised the classical machine-model of computation, but (what is less well known) already foresaw the potential of transfinite induction as a method for program verification; and of course the Herbrand–Gödel–Kleene equation calculus presented computability as a formal system of equational derivation (with "call by value" being modelled by a substitution rule which itself is a form of "cut" but at the level of terms).

That these two fields—proof and recursion—have developed side by side over the intervening seventy-five years so as to form now a cornerstone in the foundations of computer science, testifies to the power and importance of mathematical logic in transferring what was originally a body of philosophically inspired ideas and results, down to the frontiers of modern information technology. A glance through the contents of any good undergraduate text on the fundamentals of computing should lend conviction to this argument, but we hope also that some of the examples and applications in this book will support it further.

Our book is not about "technology transfer" however, but rather about a fundamental area of mathematical logic which underlies it. We would not presume to compete with "classics" in the field like Kleene's [1952], Schütte's [1960], Takeuti's [1987], Girard's [1987] or Troelstra and van Dalen's [1988], but rather we aim to complement them and extend their

range of proof-theoretic applications with a treatment of topics which reflect our own personal interests over many years and include some which have not previously been covered in textbook form. Our contribution could be seen as building on the books by Rose [1984] and Troelstra and Schwichtenberg [2000]. Thus the theory of proofs, recursions, provably recursive functions, their subrecursive hierarchy classifications, and the computational significance and application of these, will constitute the driving theme. The methods will be those now-classical ones of cut elimination, normalization, functional interpretations, program extraction and ordinal analyses, but restricted to the "small-to-medium-sized" range of mathematically significant proof systems between polynomial time arithmetic and (restricted) Π^1_1-comprehension or $\mathrm{ID}_{<\omega}$. Within this range we hope to have something new to contribute. Beyond it, the "outer limits" of ordinal analysis and the emerging connections there with large cardinal theory are presently undergoing rapid and surprising development. Who knows where that will lead?—others are far better equipped to comment.

The fundamental point of proof theory as we see it is Kreisel's dictum: a proof of a theorem conveys more information than the mere statement that it is true (at least it does if we know how to analyse its structure). In a computational context, knowledge of the truth of a "program specification"

$$\forall_{x\in\mathbb{N}}\,\exists_{y\in\mathbb{N}}\,\mathrm{Spec}(x,y)$$

tells us that there is a while-program

$$y := 0\textbf{; while } \neg\mathrm{Spec}(x,y) \textbf{ do } y := y+1\textbf{; } p := y$$

which satisfies it in the sense that

$$\forall_{x\in\mathbb{N}}\,\mathrm{Spec}(x,p(x)).$$

However, we know nothing about the complexity of the program without knowing why the specification was true in the first place. What we need is a proof! Even when we have one it might use lemmas of logical complexity far greater than Σ^0_1, and this would prevent us from analysing directly the computational structure embedded within it. So what is required is a method of reducing the proof and the applications of lemmas in it, to a "computational" (Σ^0_1) form, together with some means of measuring the cost or complexity of that reduction. The method is cut elimination or normalization and the measurement is achieved by ordinal analysis.

One may wonder why transfinite ordinals enter into the measurement of program complexity. The reason is this: a program, say over the natural numbers, is a syntactic description of a type-2 recursive functional which takes variable "given functions" g to output functions f. By unravelling the intended operation of the program according to the various function calls it makes in the course of evaluation, one constructs a tree of subcomputations, each branch of which is determined by an input number

for the function f being computed together with a particular choice of given function g. To say that the program "terminates everywhere" is to say that every branch of the computation tree ends with an output value after finitely many steps. Thus

$$\text{termination} \quad = \quad \text{well-foundedness}\,.$$

But what is the obvious way to measure the size of an infinite well-founded tree? Of course, by its ordinal height or rank!

We thus have a natural hierarchy of total recursive functionals in terms of the (recursive) ordinal ranks of their defining programs. Kleene was already aware in 1958 that this hierarchy continues to expand throughout the recursive ordinals—i.e., for each recursive ordinal α there is a total recursive functional which cannot be defined by any program of rank $< \alpha$. The "subrecursive classification problem" therefore has a perfectly natural and satisfying solution when viewed in the light of type-2 functionals, in stark contrast to the rather disappointing state of affairs in the case of type-1 functions—where "intensionality" and the question "what is a natural well-ordering?" are stumbling blocks which have long been a barrier to achieving any useful hierarchy classification of all recursive functions (in one go). Nevertheless there has been good progress in classifying subclasses of the recursive functions which arise naturally in a proof-theoretic context, and the later parts of this book will be much concerned with this.

Proofs in mathematics generally deal with abstract, "higher-type" objects. Therefore an analysis of computational aspects of such proofs must be based on a theory of computation in higher types. A mathematically satisfactory such theory has been provided by Scott [1970] and Ershov [1977] (see also Chernov [1976]). The basic concept is that of a *partial continuous functional*. Since each such can be seen as a limit of its finite approximations, we get for free the notion of a computable functional: it is given by a recursive enumeration of finite approximations. The price to pay for this simplicity is that functionals are now *partial*, in stark contrast to the view of Gödel [1958]. However, the total functionals can be defined as a subset of the partial ones. In fact, as observed by Kreisel, they form a dense subset with respect to the Scott topology. The next step is to build a theory, with the partial continuous functionals as the intended range of its (typed) variables. This is TCF, a "theory of computable functionals". It suffices to restrict the prime formulas to those built with inductively defined predicates. For instance, falsity can be defined by $\boldsymbol{F} := \mathrm{Eq}(\mathtt{ff}, \mathtt{tt})$, where Eq is the inductively defined Leibniz equality. The only logical connectives are implication and universal quantification: existence, conjunction and disjunction can all be seen as inductively defined (with parameters). TCF is well suited to reflect on

the computational content of proofs, along the lines of the Brouwer–Heyting–Kolmogorov interpretation, or more technically a realizability interpretation in the sense of Kleene and Kreisel. Moreover the computational content of classical (or "weak") existence proofs can be analyzed in TCF, by way of Gödel's [1958] Dialectica interpretation and the so-called A-translation of Friedman [1978] and Dragalin [1979]. The difference of TCF to well-established theories like Martin-Löf's [1984] intuitionistic type theory or the theory of constructions underlying the Coq proof assistant is that TCF treats partial continuous functionals as first-class citizens. Since they are the mathematically correct domain of computable functionals, it seems (to us) that this is a reasonable step to take.

Our aim is to bring these issues together as two sides of the same coin: on one the proof-theoretic aspects of computation, and on the other the computational aspects of proof. We shall try to do this in progressive stages through three distinct parts, keeping in mind that we want the book to be self-contained, orderly and fairly complete in its presentation of our material, and also useful as a reference. Thus we begin with two basic chapters on proof theory and recursion theory, followed by Chapter 3 on Gödel's theorems, providing the fundamental material without which any book with this title would be incomplete. Part 2 deals with the now fairly classical results on hierarchies of provably recursive functions for a spectrum of theories ranging between $\mathrm{I}\Delta_0(\exp)$ and $\Pi^1_1\text{-}\mathrm{CA}_0$. The point is that, just as in other areas of mathematical logic, ordinals (in our case recursive ordinals) provide a fundamental abstract mathematical scale against which we can measure and compare the logical complexity of inductive proofs and the computational complexity of recursive programs specified by them. The bridge is formed by the fast-, medium- and slow-growing hierarchies of proof-theoretic bounding functions which are quite naturally associated with the ordinals themselves, and which also "model" in a clear way the basic computational paradigms: "functional", "while-loop" and "term-reduction". We also bring out connections between fast-growing functions and combinatorial independence results such as the modified finite Ramsey theorem and the extended Kruskal theorem, for labelled trees. Part 3 develops the fundamental theory of computable functionals TCF. This is also the theory underlying the first author's proof-assistant and program-extraction system Minlog[1]. The implementation is not discussed here, but the underlying proof-theoretic ideas and the various aspects of constructive logic involved are dealt with in some detail. Thus: the domain of continuous functionals in which higher-type computation naturally arises, functional interpretations, and finally implicit complexity, where ideas developed throughout the whole book are brought to bear on certain newer weak systems with more "feasible"

[1] See http://www.minlog-system.de.

provable functions. Every chapter is intended to contain some examples or applications illustrating our intended theme: the link between proof theory, recursion and computation.

Although we have struggled with this book project over many years, we have found the writing of it more and more stimulating as it got closer to fruition. The reason for this has been a divergence of our mathematical "standpoints"—while one (S.W.) holds to a more pragmatic middle-of-the-road stance, the other (H.S.) holds a somewhat clearer and committed constructive view of the mathematical world. The difference has led to many happy hours of dispute and this inevitably may be evident in the choice of topics and their presentations which follow. Despite these differences, both authors believe (to a greater or lesser extent) that it is a rather extreme position to hold that existence is really equivalent to the impossibility of non-existence. Foundational studies—even if classically inspired—should surely investigate these positions to see what relative properties the "strong" ($\exists$) and "weak" ($\tilde{\exists}$) existential quantifiers might possess.

A brief guide to the reader. Part 1 could be the basis for a year-long graduate logic course, possibly extended by selected material from parts 2 and 3. We have endeavoured (though not with complete success) to make each chapter fairly self-contained, so that the interested reader might access any one of them directly. All succeeding chapters require a small amount of material from chapters 1 and 2, and chapters 7 and 8 rely on some concepts from chapter 6 (recursion operators, algebras and types), but otherwise all chapters can be read almost independently.

Acknowledgement. We would like to thank the many people who have contributed to this book in one way or another. They are too numerous to be listed here, but most of them appear in the references. The material in parts 1 and 2 has been used as a basis for graduate lecture courses by both authors, and we gratefully acknowledge the many useful student contributions to both the exposition and the content. Simon Huber—in his diploma thesis [2010]—provided many improvements and/or corrections to part 3. Wilfried Buchholz's work has significantly influenced part 2, and he made helpful comments on the final section of chapter 3. Our special thanks go to Josef Berger and Grigori Mints, who kindly agreed to critically read the manuscript. Thanks also to the Mathematisches Forschungsinstitut Oberwolfach for allowing us the opportunity to meet occasionally, under the Research in Pairs Programme, while the book was in its formative stages.