

Review of  
**DERIVATION AND COMPUTATION: taking the Curry-Howard correspondence  
seriously**<sup>1</sup>  
**Author: Harold Simmons**  
**Series: Cambridge Tracts in Theoretical Computer Science, Volume 51**  
**Cambridge University Press, 2000**  
**Hardcover, xxv + 384 pages, \$75.00 from the publisher**  
Reviewer: Robert J. Irwin, SUNY Oswego

## 1 Overview

The term *Curry-Howard correspondence* (also: *Curry-Howard isomorphism* or *formulae-as-types correspondence*) refers to the relationship between formal calculi for logical derivations and formal calculi for computations. For example, minimal logic — the implicational fragment of intuitionistic propositional logic — is isomorphic to the simply-typed  $\lambda$ -calculus. How so? If we associate the propositional variables of the logic with the type variables of the  $\lambda$ -calculus, the logical formulae will then correspond to simple types. Moreover, proofs in minimal logic correspond to  $\lambda$ -terms of appropriate type, and the provability of a formula corresponds to the existence of a  $\lambda$ -term of that type (i.e., to that type’s being *inhabited*). The germinal idea, due to Haskell B. Curry [1, 2] in the context of combinatory logic with type assignment, was applied to typed  $\lambda$ -calculi and popularized by William A. Howard in a manuscript widely circulated since 1969, but only published in 1980 [3].

The subtitle is a bit curious — folks have “[taken] the Curry-Howard correspondence seriously” for decades and many texts on type theory feature the correspondence. For example, the influential and oft-cited book that inspired the author to write the one under review [4] is about nothing else, and the excellent introductory text [7] devotes a chapter to the subject. Readily available lecture notes [5] treat the correspondence, and no modern textbook on the theory of programming languages (e.g, [6]) can be complete without at least mention of the correspondence.

While the correspondence between minimal logic and simply-typed  $\lambda$ -calculus is particularly transparent, it turns out that an odd assortment of background knowledge is needed to progress to isomorphisms between richer pairs of logical and computational calculi. Simmons’s text, whose thrust is pedagogical, intends to be a not-quite-one-stop-shopping experience for readers interested in the interplay between the kinds of symbol-pushing undertaken in logic and in computation.

## 2 Summary of Contents

As the author writes in the Introduction, “[t]here is nothing worse than an exercise you can’t do and have no way of finding a solution to it.” That is why, following about 200 pages of exposition, with exercises, the remainder of the text is devoted to the presentation of (mostly) fully worked solutions to *all* the preceding exercises, of which there are approximately 200.

After the Introduction comes a “Preview” chapter, which will help readers integrate the tuition they are about to receive in the next nine chapters. Only a background in basic logic is assumed — Simmons does not assume other authors have conditioned his readers for the news. Chapters on “Derivation Systems” (minimal logic via natural deduction and Hilbert-type formalisms), basic “Computation Mechanisms” (untyped combinatory logic and  $\lambda$ -calculus), “The Typed Combinator Calculus”, “The Typed  $\lambda$ -Calculus” and “Substitution Algorithms” prepare the way for the key

---

<sup>1</sup>©Robert J. Irwin, 2005

chapter on “Applied  $\lambda$ -Calculi.” In the last-mentioned chapter, the emphasis shifts to the interpretation of types as function spaces and terms as particular functions, leading to the investigation of recursion and induction over the natural numbers and Gödel’s system T [9] (the primitive recursive functionals).

Subsequent chapters cover “Multi-Recursive Arithmetic”, “Ordinals and Ordinal Notations” and, finally, “Higher Order Recursion”. The multi-recursive arithmetic chapter studies recursion in some detail, covering simultaneous recursion over multiple arguments, etc., the sort of material typically given short shrift in introductory (and many advanced) treatments of recursion theory, but featured in more specialized works like [10, 11]. Higher-order formulations of recursion are captured via the calculus  $\lambda\mathbf{G}$ , essentially Gödel’s T minus the equational reasoning. Hierarchies of number-theoretic functions are studied and their relative complexities are shown to correspond with their formulations in  $\lambda\mathbf{G}$ .

The chapter on ordinals and ordinal notations provides material often omitted from less self-contained treatments. Still, as in other chapters, not all the necessary background is provided for doing all the related exercises, though the reader is alerted when extra information is needed, and given references to the select bibliography to supply the missing pieces (e.g., a proof of the Cantor normal form theorem). The significance of the differences between ordinals and ordinal notations is clearly established. Treatment of the ordinals less than the first critical ordinal  $\epsilon_0$  is given, which allows the analysis of  $\lambda\mathbf{G}$  to be completed in the final expository chapter.

Following the exposition are nine more chapters, each containing solutions to the exercises of the corresponding expository chapter. Solutions are presented with a level of care and detail seldom encountered in other texts.

### 3 Opinion

This text is recommended for the student or researcher who’s been exposed to bits and pieces of the Curry-Howard correspondence, but wants a sharper idea of the big picture and is willing to work through the exercises to see how the details fit together. Simmons has succeeded in pulling together the main fruits of the correspondence for simple types in a single text.

Regarding style, the book is rather snappily written. It’s informal, breezy — sometimes positively jaunty — and always directly addressed to the reader. Rarely, the informality partly defeats the explanation of parts of what is, essentially, a highly formal subject. Though well thought-out overall, and elaborately typeset, the book retains some of the feel of lecture notes, from which it in fact evolved. For example, the occasional definition is oddly written, with the definiendum unclear or withheld to the last.

The short, but annotated and well-selected bibliography is sufficient to fill the gaps in the text. Still, I wouldn’t recommend the book for rank beginners, who would be well-prepared by learning the material in [7], which treats almost exclusively Curry’s simple type theory (“type-assignment”), and perhaps a good, compatible proof theory book, say [8]. These two texts combined contain no more exercises than Simmons’s, and only a fraction of them are solved.

From the Preview: “I could rationalize the choice of topics, but in the end this wouldn’t convince you if I have missed your favourite.” What I miss is Schwichtenberg’s result that the “extended” polynomials (polynomials plus the conditional) are precisely the functions definable in the simply-typed  $\lambda$ -calculus [12], which could have been presented nicely in a few short exercises.

It can’t be emphasized enough that the great thing about this book is its many well-chosen, completely solved exercises. This alone makes it a valuable text, especially for self-study.

## References

- [1] H.B. Curry. Functionality in Combinatory Logic, *Proceedings of the National Academy of the U.S.A.*, 20 (1934), 584-590.
- [2] H.B. Curry and R. Feys. *Combinatory Logic, Vol. 1*, North Holland, 1958; 2nd ed., 1968.
- [3] W.A. Howard. The formulae-as-types notion of construction, in *To H.B. Curry*, ed. J.R. Hindley and J.P. Seldin, Academic Press, UK 1980, pp. 479-490.
- [4] J-Y. Girard, Y. Lafont and P. Taylor. *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science, Volume 7, Cambridge U. Press, 1989.
- [5] M.H. Sorensen and P. Urzyczyn. *Lectures on the Curry-Howard isomorphism*. Technical Report 98/14, DIKU, Copenhagen, 1998.
- [6] B.C. Peirce. *Types and Programming Languages*, MIT Press, 2002.
- [7] J. Roger Hindley. *Basic Simple Type Theory*. Cambridge Tracts in Theoretical Computer Science, Volume 42, Cambridge U. Press, 1997.
- [8] A.S. Troelstra and H. Schwichtenberg. *Basic Proof Theory, 2nd ed.*, Cambridge Tracts in Theoretical Computer Science, Volume 43, Cambridge U. Press, 2000.
- [9] K. Gödel. Über eine bisher noch nicht unbenützte Erweiterung des finiten Standpunktes, *Dialectica* 12:280-287, 1958.
- [10] R. Péter. *Recursive Functions*. Academic Press, 1967.
- [11] H.E. Rose. *Sub-Recursion: Functions and Heirarchies*, Oxford Logic Guides, Oxford U. Press, 1984.
- [12] H. Schwichtenberg. Definierbare Funktionen im  $\lambda$ -Kalkül mit Typen, *Archiv für Mathematische Logik und Grundlagenforschung*, 17:113-114, 1975.