This book is currently out of print. Upon kind permission of the M.I.T.-Press, it is available on
ftp.ens.fr/pub/dmi/users/longo/CategTypesStructures
All references should be made to the published book.

CATEGORIES
TYPES
AND STRUCTURES

An Introduction to Category Theory for the working computer scientist

Andrea Asperti

Giuseppe Longo

FOUNDATIONS OF COMPUTING SERIES

M.I.T. PRESS, 1991

INTRODUCTION

The main methodological connection between programming language theory and category theory is the fact that both theories are essentially "theories of functions." A crucial point, though, is that the categorical notion of morphism generalizes the set-theoretical description of function in a very broad sense, which provides a unified understanding of various aspects of the theory of programs. This is one of the reasons for the increasing role of category theory in the semantic investigation of programs if compared, say, to the set-theoretic approach. However, the influence of this mathematical discipline on computer science goes beyond the methodological issue, as the categorical approach to mathematical formalization seems to be suitable for focusing concerns in many different areas of computer science, such as software engineering and artificial intelligence, as well as automata theory and other theoretical aspects of computation.

This book is mostly inspired by this specific methodological connection and its applications to the theory of programming languages. More precisely, as expressed by the subtitle, it aims at a self-contained introduction to general category theory (part I) and at a categorical understanding of the mathematical structures that constituted, in the last twenty or so years, the theoretical background of relevant areas of language design (part II). The impact on functional programming, for example, of the mathematical tools described in part II, is well known, as it ranges from the early dialects of Lisp, to Edinburgh ML, to the current work in polymorphisms and modularity. Recent applications, such as CAML, which will be described, use categorical formalization for the purposes of implementation.

In addition to its direct relevance to theoretical knowledge and current applications, category theory is often used as an (implicit) mathematical jargon rather than for its explicit notions and results. Indeed, category theory may prove useful in construction of a sound, unifying mathematical environment, one of the purposes of theoretical investigation. As we have all probably experienced, it is good to know in which "category" one is working, i.e., which are the acceptable morphisms and constructions, and the language of categories may provide a powerful standardization of methods and language. In other words, many different formalisms and structures may be proposed for what is essentially the same concept; the categorical language and approach may simplify through abstraction, display the generality of concepts, and help to formulate uniform definitions. This has been the case, for example, in the early applications of category theory to algebraic geometry.

The first part of this book should encourage even the reader with no specific interest in programming language theory to acquire at least some familiarity with the categorical way of looking at formal descriptions. The explicit use of deeper facts is a further step, which becomes easier with access to this information. Part II and some chapters in part I are meant to take this further step, at least in one of the possible directions, namely the mathematical semantics of data types and programs as objects and morphisms of categories.

We were urged to write the general introduction contained in part I, since most available books in category theory are written for the "working mathematician" and, as the subject is greatly indebted to algebraic geometry and related disciplines, the examples and motivations can be understood only by readers with some acquaintance with nontrivial facts in algebra or geometry. For most computer scientists, it is not much help in the understanding of "natural transformations" to see an involved example based on tensor products in categories of sheaves. Thus our examples will be based on elementary mathematical notions, such as the definition of monoid, group, or topological space, say, and on structures familiar for readers with some acquaintance with the tools in programming language semantics. In particular, partial orders and the various categories of domains for denotational semantics will often be mentioned or introduced, as well as basic results from computability theory. For example, we will try to present the fundamental operation of "currying" for cartesian closed categories with reference to the connection between the universal function and the g del- numbering of the partial recursive functions. Partial morphisms will be presented as a generalization of a common notion in theory of computation.

Category theory may be presented in a very abstract way: as a pure game of arrows and diagrams. It is useful to reach the point where acquaintance with the formal (essentially, equational) approach is so firm that it makes sense independently of any "structural" understanding. In this book, though, we will stress the role of structures, and we will always try to give an independent meaning to abstract notions and results. Each definition and fact will be exemplified, or even derived, from applications or structures in some way indebted to computing. However, in order to stress the role of the purely equational view, the last chapters of each part (essentially chapters 7 and 11) will be largely based on a formal, computational approach. Indeed, even if mathematically very abstract, the equational arguments turn out to be particularly relevant from a computer science perspective.

The early versions of this book grew out of two graduate courses taught by Longo in Pisa, in 1984/85, and at Carnegie Mellon University, in 1987/88. Then the book was entirely revised under the influence of Asperti's work for his Ph.D. dissertation. In particular, chapters 7 and 11, the technically most difficult, are part of his dissertation.

We are indebted to several people. The joint work with Simone Martini and Eugenio Moggi in several papers directly influenced many chapters. Moreover, Eugenio suggested, in handwritten notes and electronic mail messages, the basic ideas for the categorical understanding of polymorphism via internal categories and realizability toposes. Their mathematical insights and suggestions also influenced other parts of the book.

We must acknowledge the influence on our approach of the ideas and work of Dana Scott and Gordon Plotkin, who also encouraged us and made comments on early drafts. Pino Rosolini helped us with comments and many suggestions. Jean Yves Girard and Yves Lafont brought to our attention the tidy categorical meaning of linear logic and its applications to computing. Roberto Amadio and many students helped us by detecting errors and incompleteness in the presentation. We are looking forward to aknowledge the readers who will detect the remaining errors.

The first draft of this book was completed while the authors were visiting Carnegie Mellon University, in 1987/88. Longo would like to thank the Computer Science Dept. of CMU for its very generous hospitality while he was teaching there that academic year. The circulation of the draft, its complete revision, and the writing of the final version of the book have been made possible by the Joint Collaboration Contract ST2J-0374-C (EDB) of the European Economic Community and by the Italian CNR "Stanford-grant" #89.00002.26. The authors would like to thank INRIA, Rocquencourt, for a postdoc granted to Asperti while completing this work and l'Ecole Normale Supérieure, Paris, for inviting Longo to teach a graduate course in 1989/90 based partly on this book.

TABLE OF CONTENTS

PART I: Categories and Structures

PART II: Types as Objects