

PREFACE

Why another book on Category Theory?

In the past ten years, several books have been published on Category Theory either by Computer Scientists or having Computer Scientists as a target audience (e.g. [6,12,22,91,107], to which we add a precious collection of little gems [92] should be added). Isn't the Working Computer Scientist spoilt with choice?

Although each of the above mentioned books presents an approach of its own, there is one aspect in common in their view of Computer Science: the analogy between arrows (morphisms) and (classes of) computations. This "type-theoretic" or "functional" approach corresponds to a view of Computer Science as a Science of Computation, i.e. a discipline concerned with the study of computational phenomena where the focus is on the nature and organisation of computations.

However, there is another view of Computer Science where the focus is, instead, on the development of computer programs or systems. This is the approach that supports, for instance, Software Engineering. From this point of view, arrows capture not computational phenomena, or abstractions thereof, but relationships between programs, or abstractions of programs, that arise in the development of computer systems: for instance, refinement of higher-level specifications into executable programs [101,106], and superposition of new features over existing systems [71].

Not surprisingly, this same difference in the points of view can be found when Logic (instead of Category Theory) is taken as a mathematical domain for formalising aspects of Computer Science. The "computations as proofs" paradigm is the one that corresponds to the "classical" application of Category Theory. Terms of the logic correspond to objects in a category of programs whose morphisms capture (partial) computations. From a logical point of view, the perspective that we take in this book is not centred on terms but on theories as system specifications. Morphisms then capture what in Logic is known as "interpretations between theories", the cornerstone for the formalisation of refinement in program development and other operations on specifications and system designs [17,81,82,83,84,104,105,106].

Category Theory can also be presented as the branch of Mathematics that, *par excellence*, addresses "structure". As the introduction will try to motivate, this is because Category Theory causes structure to emerge from relationships between objects as captured by arrows, and not extensionally as in Set Theory. Indeed, the term *morphism* often used for arrow in Category Theory, has in its etymology the notion of "preservation of form". What these structures are, or mean, is up to the "user". Hence, in the "classical" approach, we find applications of Category Theory that address the structure of computations. In the approach that is taken in this book, the structures that will be addressed are the ones that capture modularisation principles in software development, in particular, those that have been emerging in the guise of what has become known as Software Architectures [49].

The practical difference between the two approaches in what concerns Category Theory in general, and this book in particular, is that the reader will not find as many references to Algebraic Topology or related fields of Mathematics as applied, for instance, to Domain Theory. Although this book is still “mathematical”, the Software Practitioner will find the mathematics applied to objects of its day-to-day concerns: programs, object classes, specifications, designs, etc.

This approach can be also situated as belonging to the class of applications of Category Theory to General Systems Theory, namely in the tradition initiated in [51,52,63], an area of Science that, as the name indicates and the introduction will try to motivate, encompasses more than computational systems in the traditional sense. Through books aimed at wider audiences like [70], a unifying view of complex systems as they arise in disparate areas like Physics, Biology, Social Sciences, Economics and, yes, Informatics, has started to emerge (pun intended) which is a clear indication of new levels of maturity in Science in general and Informatics in particular. Hence, one of the purposes of this book is to help Computing scientists and Software practitioners acquire formal tools that will enable them to follow and participate in this “new” culture.

A trait that is common to all these areas is a view of complex systems as communities of interacting, simpler, autonomous entities. Whereas, in areas like Biology or Social Sciences, the notion of “community” is intrinsic, its use in areas like Software Engineering is more artificial and is normally identified with methods and development techniques that, in the past few years, have attempted to tackle complexity by borrowing the organisational principles that can be recognised in such “natural” communities. Object-oriented modelling, agent-based programming, and component-based development all make use, in one way or another, and with different emphasis, of this analogy. This brings us to the second word in the name of this book, the one that identifies its second part.

CommUnity is the name of a language for parallel program design that is similar to Unity [19] but adopts instead an interaction model that places it in the realm of these more general and unifying approaches to systems. It addresses in particular the most recent trend – *service-oriented* software development, an (r)evolution of the popular object-object oriented modelling techniques for the “Internet-age” or what is becoming known as the “real-time” or “now” Economy. The distinctive feature of this new trend is in the emphasis that it puts in the externalisation and explicit modelling of interactions as first-class citizens so that systems can be more easily reconfigured, in run-time, and without interruption of vital services. These characteristics match, precisely, features that are intrinsic to Category Theory, namely those that distinguish it from Set Theory. This is why, even if a substantial part of this book is illustrated with many examples borrowed from Software Engineering practice, we decided to devote three chapters to the application of Category Theory to CommUnity and its relationship to Software Architectures. This will provide an opportunity for the reader to see the majority of the basic concepts and techniques of Category Theory applied in an integrated and systematic way. At the same time, the reader will be able to appreciate how far one can go in formalising software development methods and techniques in mathematical frameworks, which is essential for a mature Engineering discipline and, in my opinion, our responsibility as scientists.

How this book came into being and whom you should blame for it

Mentioning the connections between Category Theory and General Systems Theory is a good opportunity to give due credit to Joseph Goguen for the profound inspiration that his work has instilled, a sentiment that I know is shared by many other researchers in Computing Science. He has expressed his own views on the applica-

tions of Category Theory to Computing in several publications, most notably in [56], which include detailed summaries of technical results that we all have found very useful when categorical approaches were still regarded, at best, as "exotic" [59,60,65,103]. All of us regret that this material has never found its way to a textbook. Because it is not the aim of this book to fill this gap, the reader is strongly encouraged to consult this rich legacy at his or her own pace, bearing in mind that the list of references that is provided at the end is far from being complete.

Completeness is, in fact, a concern that has remained largely alien to my research agenda. (This observation is intended to make some faces smile but you can take it literally.) This book is more about a personal experience than the output of a rational process of identifying "the" or "a" complete categorical kernel that Software Engineers can use as a toolbox. The only justification for the inclusion of many concepts and constructions is that they were of help to me, either technically or aesthetically, making it "likely" that they will be directly useful for other people "like me". The exclusion of many other, even very basic ones¹, can be justified by the officious disclaimer "the line has to be drawn somewhere" but, most of the time, the reason is that I never really stumbled upon them in my daily routine or simply that I have not developed an understanding about them that is deep enough to add any value to what can be found in other books.

This personal experience has gone through well identifiable periods, each of which is associated with a different focus of interest in Computing and a group of people with which I worked directly and whose contributions I would like to acknowledge. My first contact with Category Theory was when I was studying Mathematics as an undergraduate at the University of Lisbon, and Prof. Furtado Coelho challenged the wrath of my fellow students, and his fellow staff, by including this most exotic, difficult and useless of subjects in the curriculum of Algebra II. It was a total revelation: sheer simplicity, economy of means, and elegance! Applications to Computing came a year later through the study of Goguen and Burstall's Theory of Institutions as a means of formalising Conceptual Modelling and Knowledge Representation Approaches, under the supervision and in collaboration with Amílcar and Cristina Sernadas. This is when things started to get serious.

In 1988, I started what has been a very rewarding collaboration with Tom Maibaum. During the three years I spent at Imperial College, we developed a categorical approach to object-oriented development based on temporal logic specifications, a marriage between my previous work with institutions and the ideas of Tom and Paulo Veloso on the nature of specifications in system development. Their contribution permeates the material that will be exposed in a way that cannot be referenced in the same way as a technical result. I am very fortunate to be able to keep exchanging ideas and experiences with them: there are always hidden subtleties that only come to the surface when you are challenged by people like them and required to scratch the innermost levels of your understanding to satisfy their curiosity.

During this same time, Félix Costa in Lisbon explored the categorical semantics of objects from the point of view of algebraic models of concurrency. When I returned to Lisbon in 1992, we brought it all together! These were very exciting and rewarding times. My collaboration with Félix provided much of the inspiration that led to my own understanding of the application of Category Theory to systems modelling. Although some specific contributions are acknowledged with references to his work, it would be unfair to reduce his contribution to this book to those occasions.

¹ Yes, I know that I will not be forgiven for having left out many "must-haves" such as the Yoneda lemma, cartesian-closed categories, topoi, monads, and so on.

The next phase is devoted to the (then) emerging field of Software Architectures. It is centred on a language – CommUnity – that I developed together with Georg Reichwein and Tom Maibaum in an initial period, and later on with my students Antónia Lopes and Michel Wermelinger. It started as a proof of concept, showing that Goguen’s categorical approach could be applied to Parallel Program Design in the style of Unity [19] and Interacting Processes [47]. Later on, it evolved into a prototype language for architectural modelling, a process that led me to understand many concepts that, until then, were blurred by the poor expressive power of the formalisms with which we had been working (unfortunately, not many people have learnt from these mistakes...): non-determinism vs under-specification, refinement vs composition, and the role of signatures in separating computation and coordination. Some of this is revealed in part three of this book, but you will have to wait for another book to have the full story!

Although this “architectural” period is still very much alive (which does not mean that the other are already dead), another step in this evolution process has just taken place: the realization that Category Theory provides a perfect fit for what is required to support service-oriented software development, for instance in the sense of Web Services. But this step is so recent that, in fairness, I cannot acknowledge/blame anybody in particular for it... Nevertheless, it is unlikely that it would have happened so soon, or at all, if I had not accepted the challenge that Luís Andrade presented me with for working with ATX Software SA in putting these “theories” into “practice”... This has been a very rewarding process that has given me the opportunity to understand the implications of many of the structures and mechanisms that are intrinsic to Category Theory. I hope that I have managed to permeate this understanding in the way the material is exposed in the book.

This is probably why this book is being finalised now and only now: during each of the periods I mentioned, a book was planned and several parts were written... It is only now that the work of so many people has contributed to the contents that I can safely write it on my own without feeling guilty for excluding anybody in particular from co-authoring it.

It so happens that the last thrust in writing this book was made during my first year at the University of Leicester, a renowned address for research in Category Theory and its applications to Computer Science. Although I can honestly assure the reader that the decision to join Leicester was not for the advantages of promoting this book, it is certainly a privilege for the book to be bear this affiliation!

Finally, I should thank all the colleagues and students who have tread with me the paths that you can choose to follow in this book. The opportunity to discuss and lecture on many of the topics that will be covered contributed decisively in helping me reach the level of maturity that made me decide that this book could be written. The feedback I received from tutorials presented at events such as ECOOP, ETAPS, FME, OOPSLA and TOOLS made me decide that this book should also be written.

I would like to thank in particular the University of Lisbon and the Technical University of Lisbon for the opportunities that they gave me to lecture much of the material covered in the first two parts, both at undergraduate and postgraduate level. These parts of the book were also covered in lectures given at the University of Coimbra, the Institute for Languages and Administration in Lisbon (ISLA), the Federal University of Rio Grande do Sul (Brazil), the 1992 Advanced School on Artificial Intelligence organised by J.Falcão e Cunha in the Azores, and the IX School of Computing organised by Ruy Queiroz in Recife (Brazil).

I am particularly grateful to Ugo Montanari for the invitation to lecture a 20 hours course on CommUnity and Software Architectures (part three of this book) as part of a postgraduate programme of the University of Pisa, and to Roland Backhouse and Jeremy Gibbons for the opportunity to lecture the same material at the 2002 Summer School on Generic Programming. Jeremy deserves a special thanks for sending me lots of comments and challenging questions!

Special acknowledgements

Although the previous paragraphs have given me the opportunity to acknowledge the contributions of a number of people and institutions, there are some specific colleagues to whom I would like to express my deepest gratitude for direct contributions to this book:

Félix Costa: a significant part of the material covered in part two was developed jointly with him as reported in [32,33]. As already mentioned, much of my own understanding of Category Theory and its role in Computing Science grew up from discussions with him.

Tom Maibaum: his encouragement and support in the earlier phases of the production of the book were decisive, including the setting up of the format.

Antónia Lopes and *Michel Wermelinger*: the fact that part three of this book was essentially extracted from [37,78] is a good indication of how important and extensive their contribution has been. CommUnity as we know it today is as much theirs as it is mine.

Uwe Wolter: he had the courage to use an early draft to support part of a course that he gave in 2002/2003 in Bergen and, as a result, I received loads of feedback, which was invaluable for the final tuning of the material and the way it is presented.

Finally, I would like to thank the EPSRC for an eight-month visiting fellowship at King's College London in 1999, during which most of the book was produced, and to Janet Maibaum for her help in setting Microsoft Word up to the job²!

Leicester, September 2003

² Yes, this book is a proof that Category Theory is not reserved to users of a well-known typesetting system that I will not name... And this remark is not meant as a recommendation for the products developed by a company that I have already named...

