The textbook derives from the well-known identically titled volume published in 2009: two younger co-authors have brought their experience to enrich and streamline the book, without dulling its original structure. The selection of materials and the presentation style have not substantially changed. The book reflects many years of teaching compiler courses and of doing research on formal language theory and formal methods, on compiler and language technology, and to a lesser extent on natural language processing. The more important change concerns the central topic of language parsing. It is a completely new, systematic, and unified presentation of the most important parsing algorithms, including also parallel parsing.

Goals

In the turmoil of information technology developments, the subject of the book has kept the same fundamental principles since half a century, and has preserved its conceptual importance and practical relevance. This state of affairs in a topic that is central to computer science and is based on established principles, might lead some people to believe that the corresponding textbooks are by now consolidated, much as the classical books on mathematics and physics. In reality, this is not the case: there exist fine classical books on the mathematical aspects of language and automata theory, but for what concerns the application to compiling, the best books are sort of encyclopedias of algorithms, design methods, and practical tricks used in compiler design. Indeed, a compiler is a microcosm, and features many different aspects ranging from algorithmic wisdom to computer hardware. As a consequence, the textbooks have grown in size, and compete with respect to their coverage of the last developments on programming languages, processor architectures and clever mappings from the former to the latter.

To put things in order, it is better to separate such complex topics into two parts, basic and advanced, which to a large extent correspond to the two subsystems that make a compiler: the user-language specific front-end, and the machine-language specific back-end. The basic part is the subject of this book. It covers the principles and algorithms to be used for defining the syntax of languages and for implementing simple translators. It does not include: the specialized know-how needed for various classes of programming languages (imperative, functional, object oriented, etc.), the computer architecture related aspects, and the optimization methods used to improve the machine code produced by the compiler.

Organization and Features  In other textbooks the bias towards practical aspects has reduced the attention to fundamental concepts. This has prevented their authors from taking advantage of the improvements and simplifications made possible by decades of extensive use, and from avoiding the irritating variants and repetitions that are found in the original papers. Moving from these premises, we decided to present, in a simple minimalist way, the principles and methods used in designing language syntax and syntax-directed translators.

Chapter 2 covers regular expressions and context-free grammars, with emphasis on the structural adequacy of syntactic definitions and a careful analysis of ambiguity and how to avoid it.

Chapter 3 presents finite-state recognizers and their conversion back and forth to regular expressions and grammars.

Chapter 4 presents push-down machines and parsing algorithms, focusing attention on the $LL$, $LR$ and Earley methods. We have substantially improved the standard presentation of such algorithms, by unifying the concepts and notations used in various approaches, and by extending the method coverage with a reduced definitional apparatus. An example that expert readers and instructors should appreciate, is the unification of the top-down ($LL$) and bottom-up ($LR$) parsing algorithms, as well as the tabular (Early) one, within a novel practical framework. In this way, the effort and space spared have made room for advanced methods typically not present in similar textbooks. First, our parsing algorithms apply to the Extended BNF grammars, which are the de facto standard in the language reference manuals. Second, we provide a parallel parsing algorithm that takes advantage of the many processing units of modern microprocessors, to speed-up the analysis of large files.

The book is not restricted to syntax: Chap. 5, the last one, studies translations, semantic functions (attribute grammars), and the static program analysis by data flow equations. This provides a comprehensive understanding of the compilation process, and covers the essential elements of a syntax-directed translator.

The presentation is illustrated by many small yet realistic examples and pictures, to ease the understanding of the theory and the transfer to application. Theoretical models of automata, transducers and formal grammars are extensively used, whenever practical motivations exist, without insisting too much on their formal definition. Algorithms are described in a pseudo-code to avoid the disturbing details of a programming language, yet they are straightforward to convert to executable procedures.

This book should be welcome by those willing to teach or to learn the essential concepts of syntax-directed compilation, without the need of relying on software tools and implementations. We believe that learning by doing is not always the best approach, and that over-commitment to practical work may sometimes obscure the conceptual foundations. In the case of formal languages, the elegance and simplicity of the underlying theory allows the students to acquire the fundamental paradigms of language structures, to avoid pitfalls such as ambiguity, and to adequately map structure to meaning. In this field, the relevant algorithms are simple enough to be practiced by paper and pencil. Of course, students should be encouraged to enroll in a hands-on laboratory and to experiment syntax-directed tools (like flex and bison) on realistic cases.
Intended Audiences This is primarily a textbook targeted to graduate (or upper-division undergraduate) students in computer science or computer engineering. We list as prerequisites: familiarity with some programming language models and with algorithm design; and the ability to use elementary mathematical and logical notation. If the reader or student has already taken a course on theoretical computer science including a few elements of formal language and automata theory, the time and effort needed to learn the first chapters can be reduced. But it is fair to say that this book does not compete with the many available books on the theory of formal languages and computation: we usually do not include mathematical proofs of properties, and we rely instead on examples and informal arguments. Yet mathematically oriented readers will find here many motivating examples and applications.

A large collection of problems and solutions complementing the numerous examples in the book is available on the authors’ course web site at Politecnico di Milano. Similarly, a comprehensive set of lecture slides is also available on the course web site.

The Authors Thank The colleagues who have taught this book to computer engineering students, Giampaolo Agosta and Licia Sbattella; the Italian National Research Group on Formal Languages, in particular Antonio Restivo, Alessandra Cherubini, Dino Mandrioli, Matteo Pradella and Pierluigi San Pietro; our past and present Ph.D. students and teaching assistants, Alessandro Barenghi, Marcello Bersani, Andrea Di Biagio, Simone Campanoni, Silvia Lovergine, Michele Scandale, Ettore Speziale, Martino Sykora and Michele Tartara. We also acknowledge the support of ST Microelectronics Company for R.&D. projects on compilers for advanced microprocessor architectures.

The first author remembers the late Antonio Grasselli, a pioneer of computer science studies, who first fascinated him with a subject combining linguistic, mathematical, and technological aspects.

Milan, Italy
July 2013

Stefano Crespi Reghizzi
Luca Breveglieri
Angelo Morzenti
Formal Languages and Compilation
Crespi Reghizzi, S.; Breveglieri, L.; Morzenti, A.
2013, XII, 399 p. 171 illus., 1 illus. in color., Hardcover
ISBN: 978-1-4471-5513-3