Preface

Parsing (syntactic analysis) is one of the best understood branches of computer science. Parsers are already being used extensively in a number of disciplines: in computer science (for compiler construction, database interfaces, self-describing data-bases, artificial intelligence), in linguistics (for text analysis, corpora analysis, machine translation, textual analysis of biblical texts), in document preparation and conversion, in typesetting chemical formulae and in chromosome recognition, to name a few; they can be used (and perhaps are) in a far larger number of disciplines. It is therefore surprising that there is no book which collects the knowledge about parsing and explains it to the non-specialist. Part of the reason may be that parsing has a name for being "difficult". In discussing the Amsterdam Compiler Kit and in teaching compiler construction, it has, however, been our experience that seemingly difficult parsing techniques can be explained in simple terms, given the right approach. The present book is the result of these considerations.

This book does not address a strictly uniform audience. On the contrary, while writing this book, we have consistently tried to imagine giving a course on the subject to a diffuse mixture of students and faculty members of assorted faculties, sophisticated laymen, the avid readers of the science supplement of the large newspapers, etc. Such a course was never given; a diverse audience like that would be too uncoordinated to convene at regular intervals, which is why we wrote this book, to be read, studied, perused or consulted wherever or whenever desired.

Addressing such a varied audience has its own difficulties (and rewards). Although no explicit math was used, it could not be avoided that an amount of mathematical thinking should pervade this book. Technical terms pertaining to parsing have of course been explained in the book, but sometimes a term on the fringe of the subject has been used without definition. Any reader who has ever attended a lecture on a non-familiar subject knows the phenomenon. He skips the term, assumes it refers to something reasonable and hopes it will not recur too often. And then there will be passages where the reader will think we are elaborating the obvious (this paragraph may be one such place). The reader may find solace in the fact that he does not have to doodle his time away or stare out of the window until the lecturer progresses.

On the positive side, and that is the main purpose of this enterprise, we hope that by means of a book with this approach we can reach those who were dimly aware of the existence and perhaps of the usefulness of parsing but who thought it would forever

be hidden behind phrases like:

Let $\mathfrak{P}$ be a mapping $V_N \xrightarrow{\mathfrak{P}} 2^{(V_N \cup V_T)^*}$ and $\mathfrak{S}$ a homomorphism ...

No knowledge of any particular programming language is required. The book contains two or three programs in Pascal, which serve as actualizations only and play a minor role in the explanation. What is required, though, is an understanding of algorithmic thinking, especially of recursion. Books like *Learning to program* by Howard Johnston (Prentice-Hall, 1985) or *Programming from first principles* by Richard Bornat (Prentice-Hall 1987) provide an adequate background (but supply more detail than required). Pascal was chosen because it is about the only programming language more or less widely available outside computer science environments.

The book features an extensive annotated bibliography. The user of the bibliography is expected to be more than casually interested in parsing and to possess already a reasonable knowledge of it, either through this book or otherwise. The bibliography as a list serves to open up the more accessible part of the literature on the subject to the reader; the annotations are in terse technical prose and we hope they will be useful as stepping stones to reading the actual articles.

On the subject of applications of parsers, this book is vague. Although we suggest a number of applications in Chapter 1, we lack the expertise to supply details. It is obvious that musical compositions possess a structure which can largely be described by a grammar and thus is amenable to parsing, but we shall have to leave it to the musicologists to implement the idea. It was less obvious to us that behaviour at corporate meetings proceeds according to a grammar, but we are told that this is so and that it is a subject of socio-psychological research.

Acknowledgements
We thank the people who helped us in writing this book. Marion de Krieger has retrieved innumerable books and copies of journal articles for us and without her effort the annotated bibliography would be much further from completeness. Ed Keizer has patiently restored peace between us and the pic|tbl|eqn|psfig|troff pipeline, on the many occasions when we abused, overloaded or just plainly misunderstood the latter. Leo van Moergestel has made the hardware do things for us that it would not do for the uninitiated. We also thank Erik Baalbergen and Frans Kaashoek for their critical remarks and contributions. The rose at the end of Chapter 2 is by Arwen Grune. Ilana and Lily Grune typed parts of the text on various occasions.

We thank the Faculteit Wiskunde en Informatica of the Vrije Universiteit for the use of the equipment.

In a wider sense, we extend our thanks to the hundreds of authors who have been so kind as to invent scores of clever and elegant algorithms and techniques for us to exhibit. We hope we have named them all in our bibliography.

Dick Grune
Ceriel J.H. Jacobs
Amstelveen/Amsterdam, July 1990

1

Introduction

Parsing is the process of structuring a linear representation in accordance with a given grammar. This definition has been kept abstract on purpose, to allow as wide an interpretation as possible. The "linear representation" may be a sentence, a computer program, a knitting pattern, a sequence of geological strata, a piece of music, actions in ritual behaviour, in short any linear sequence in which the preceding elements in some way restrict[†] the next element. For some of the examples the grammar is well-known, for some it is an object of research and for some our notion of a grammar is only just beginning to take shape.

For each grammar, there are generally an infinite number of linear representations ("sentences") that can be structured with it. That is, a finite-size grammar can supply structure to an infinite number of sentences. This is the main strength of the grammar paradigm and indeed the main source of the importance of grammars: they summarize succinctly the structure of an infinite number of objects of a certain class.

There are several reasons to perform this structuring process called parsing. One reason derives from the fact that the obtained structure helps us to process the object further. When we know that a certain segment of a sentence in German is the subject, that information helps in translating the sentence. Once the structure of a document has been brought to the surface, it can be converted more easily.

A second is related to the fact that the grammar in a sense represents our understanding of the observed sentences: the better a grammar we can give for the movements of bees, the deeper our understanding of them is.

A third lies in the completion of missing information that parsers, and especially error-repairing parsers, can provide. Given a reasonable grammar of the language, an error-repairing parser can suggest possible word classes for missing or unknown words on clay tablets.

[†] If there is no restriction, the sequence still has a grammar, but this grammar is trivial and uninformative.

1.1 PARSING AS A CRAFT

Parsing is no longer an arcane art; it has not been so since the early 70's when Aho, Ullman, Knuth and many others put various parsing techniques solidly on their theoretical feet. It need not be a mathematical discipline either; the inner workings of a parser can be visualized, understood and modified to fit the application, with not much more than cutting and pasting strings.

There is a considerable difference between a mathematician's view of the world and a computer-scientist's. To a mathematician all structures are static: they have always been and will always be; the only time dependence is that we just haven't discovered them all yet. The computer scientist is concerned with (and fascinated by) the continuous creation, combination, separation and destruction of structures: time is of the essence. In the hands of a mathematician, the Peano axioms create the integers without reference to time, but if a computer scientist uses them to implement integer addition, he finds they describe a very slow process, which is why he will be looking for a more efficient approach. In this respect the computer scientist has more in common with the physicist and the chemist; like these, he cannot do without a solid basis in several branches of applied mathematics, but, like these, he is willing (and often virtually obliged) to take on faith certain theorems handed to him by the mathematician. Without the rigor of mathematics all science would collapse, but not all inhabitants of a building need to know all the spars and girders that keep it upright. Factoring off certain detailed knowledge to specialists reduces the intellectual complexity of a task, which is one of the things computer science is about.

This is the vein in which this book is written: parsing for anybody who has parsing to do: the compiler writer, the linguist, the data-base interface writer, the geologist or musicologist who want to test grammatical descriptions of their respective objects of interest, and so on. We require a good ability to visualize, some programming experience and the willingness and patience to follow non-trivial examples; there is nothing better for understanding a kangaroo than seeing it jump. We treat, of course, the popular parsing techniques, but we will not shun some weird techniques that look as if they are of theoretical interest only: they often offer new insights and a reader might find an application for them.

1.2 THE APPROACH USED

This book addresses the reader at at least three different levels. The interested non-computer scientist can read the book as "the story of grammars and parsing"; he or she can skip the detailed explanations of the algorithms: each algorithm is first explained in general terms. The computer scientist will find much technical detail on a wide array of algorithms. To the expert we offer a systematic bibliography of over 400 entries, which is intended to cover all articles on parsing that have appeared in the readily available journals. Each entry is annotated, providing enough material for the reader to decide if the referred article is worth reading.

No ready-to-run algorithms have been given, except for the general context-free parser of Chapter 12. The formulation of a parsing algorithm with sufficient precision to enable a programmer to implement and run it without problems requires a considerable supporting mechanism that would be out of place in this book and in our experience does little to increase one's understanding of the process involved. The popular methods are given in algorithmic form in most books on compiler construction. The

less widely used methods are almost always described in detail in the original publication, for which see Chapter 13.

1.3 OUTLINE OF THE CONTENTS

Since parsing is concerned with sentences and grammars and since grammars are themselves fairly complicated objects, ample attention is paid to them in Chapter 2. Chapter 3 discusses the principles behind parsing and gives a classification of parsing methods. In summary, parsing methods can be classified as top-down or bottom-up and as directional or non-directional; the directional methods can be further distinguished into deterministic and non-deterministic. This scheme dictates the contents of the next few chapters. In Chapter 4 we treat non-directional methods, including Unger and CYK. Chapter 5 forms an intermezzo with the treatment of finite-state automata, which are needed in the subsequent chapters. Chapters 5 through 9 are concerned with directional methods. Chapter 6 covers non-deterministic directional top-down parsers (recursive descent, Definite Clause Grammars), Chapter 7 non-deterministic directional bottom-up parsers (Earley). Deterministic methods are treated in Chapters 8 (top-down: LL in various forms) and 9 (bottom-up: LR, etc.). A combined deterministic/non-deterministic method (Tomita) is also described in Chapter 9. That completes the parsing methods per se.

Error handling for a selected number of methods is treated in Chapter 10. The comparative survey of parsing methods in Chapter 11 summarizes the properties of the popular and some less popular methods. Chapter 12 contains the full code in Pascal for a parser that will work for any context-free grammar, to lower the threshold for experimenting.

1.4 THE ANNOTATED BIBLIOGRAPHY

The annotated bibliography is presented in Chapter 13 and is an easily accessible supplement of the main body of the book. Rather than listing all publications in alphabetic order, it is divided into fourteen named sections, each concerned with a particular aspect of parsing; inside the sections, the publications are listed chronologically. An author index replaces the usual alphabetic list. The section name plus year of publication, placed in brackets, are used in the text to refer to an author's work. For instance, the annotated reference to Earley's publication of the Earley parser [CF 1970] can be found in the section CF at the position of the papers of 1970. Since the name of the first author is printed in bold letters, the actual reference is then easily located.