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 behavior, in short any linear sequence in which the preceding elements in some way restrict[1] 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 is the subject, that information helps in understanding or translating the sentence. Once the structure of a document has been brought to the surface, it can be converted more easily.

A second reason 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 is of them.

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.

The reverse problem — given a (large) set of sentences, find the/a grammar which produces them — is called *grammatical inference*. Much less is known about it than about parsing, but progress is being made. The subject would require a complete

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

book. Proceedings of the International Colloquiums on Grammatical Inference are published as Lecture Notes in Artificial Intelligence by Springer.

1.1 Parsing as a Craft

Parsing is no longer an arcane art; it has not been so since the early 1970s 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 little 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 have not 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 them, he cannot do without a solid basis in several branches of applied mathematics, but, like them, 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 out 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 database interface writer, the geologist or musicologist who wants 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 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

1700 entries. The printed book holds only those entries referenced in the book itself; the full list is available on the web site of this book. All entries in the printed book and about two-thirds of the entries in the web site list come with an annotation; this annotation, or summary, is unrelated to the abstract in the referred article, but rather provides a short explanation of the contents and enough material for the reader to decide if the referred article is worth reading.

No ready-to-run algorithms are given, except for the general context-free parser of Section 17.3. The formulation of a parsing algorithm with sufficient precision to enable a programmer to implement and run it without problems requires a considerable support 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 18.

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 ones. This situation 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 6 through 10 are concerned with directional methods, as follows. 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 methods). Chapter 10 covers non-canonical parsers, parsers that determine the nodes of a parse tree in a not strictly top-down or bottom-up order (for example left-corner). Non-deterministic versions of the above deterministic methods (for example the GLR parser) are described in Chapter 11.

The next four chapters are concerned with material that does not fit the above framework. Chapter 12 shows a number of recent techniques, both deterministic and non-deterministic, for parsing substrings of complete sentences in a language. Another recent development, in which parsing is viewed as intersecting a context-free grammar with a finite-state automaton is covered in Chapter 13. A few of the numerous parallel parsing algorithms are explained in Chapter 14, and a few of the numerous proposals for non-Chomsky language formalisms are explained in Chapter 15, with their parsers. That completes the parsing methods per se.

Error handling for a selected number of methods is treated in Chapter 16, and Chapter 17 discusses practical parser writing and use.

1.4 The Annotated Bibliography

The annotated bibliography is presented in Chapter 18 both in the printed book and, in a much larger version, on the web site of this book. It is an easily accessible and essential supplement of the main body of the book. Rather than listing all publications in author-alphabetic order, the bibliography is divided into a number of named sections, each concerned with a particular aspect of parsing; there are 25 of them in the printed book and 30 in the web bibliography. Within the sections, the publications are listed chronologically. An author index at the end of the book replaces the usual alphabetic list of publications. A numerical reference placed in brackets is used in the text to refer to a publication. For example, the annotated reference to Earley's publication of the Earley parser is indicated in the text by [14] and can be found on page 578, in the entry marked 14.