PROLOG AND NATURAL-LANGUAGE ANALYSIS (CSLI LECTURE NOTES 10)

Fernando C. N. Pereira, and Stuart M. Shieber (SRI International)

Center for the Study of Language and Information, Stanford University, Stanford, CA 1987, viii + 260 pp.
Distributed by the University of Chicago Press
ISBN 0-937073-17-2, $28.95 (hb); ISBN 0-937073-18-0, $13.95 (sb)

Reviewed by
Patrick Saint-Dizier
IRISA-INRIA

I have often met linguists and computational linguists willing to read a short, simple, and well-documented textbook about the use of Prolog for natural-language understanding. I feel that this well-written book mainly devoted to Prolog and logic grammars to a large extent meets the challenge. I will here summarize and comment on each chapter, and conclude by pointing out aspects I would have liked to see included in this introductory book.

Besides the presentation of concepts and examples, each chapter contains several exercises left to the reader and exhaustive bibliographic notes. I only regret that no answers to the exercises are provided as is usually the case for textbooks. These exercises are often nontrivial for the beginner and commented answers would have made them much more attractive and useful. Another interesting point is the internal structure of each chapter: two themes (Prolog and logic grammars) are tackled in a parallel and progressive way.

Chapter 1 is a very short introduction to logic programming, Prolog, and natural language. It also includes a list of prerequisites (which does not constitute a real constraint for a novice in the area) and some general historical material about the field and related fields.

Chapter 2 begins with an introduction to Prolog, viewed as a means to encode knowledge in a declarative way. The presentation is simple and classical. Section 2.3, about the logic of Prolog, is probably the best presentation of the topic I have ever seen, apart from the excellent book *The Art of Prolog* (Sterling and Shapiro 1986). To conclude this brief introduction, a problem section deals with the representation of semantic networks. This chapter ends by the development of an implementation method for context-free grammars in Prolog. A trace of the parsing of a short sentence helps the user to understand how the proof procedure of Prolog is used as a parsing mechanism.

Chapter 3, entitled "Pure Prolog: Theory and application," observes the same structure as the previous chapter. It first introduces more complex concepts, techniques, and data-structures of Prolog, illustrated by typical short programs, most of them belonging to the classics of the field. Then, more material about formal aspects of Prolog (e.g., substitutions and unification, resolution and the semantics of Prolog) is given. Finally, definite-clause grammars (DCGs) are introduced. They constitute the basic form of logic grammars from which several formalisms have been developed, motivated either by linguistic considerations or by the need of providing grammar writers with more transparent formulations of grammars. DCGs have themselves partly emerged from metamorphosis grammars (Colmerauer 1978), which is a general framework to deal with type-0 grammars. Besides a basic implementation of DCGs, the authors introduce fundamental techniques to build parsing trees, to deal with PP-attachment and to express different kinds of agreements.

Chapter 4 is divided into two parts: some very basic principles for semantic interpretation are given, and then the syntactic coverage of the grammar given in the previous chapter is extended. The first section of this chapter shows how the basic principles of computational semantics are dealt with in Prolog by means of logical variables. The authors then focus on quantified noun phrases and quantifier scoping. Although the programming techniques given in this latter section are of much interest, the idea of generating all possible quantifier scopings without any further refinements seems to me to be a little too superficial, given the actual state of the art. Next, the syntactic coverage of the previous grammar is extended. Of particular interest is the way filler-gap dependencies are expressed in an efficient way for simple linguistic constructions. The technique of difference lists is then introduced to permit an efficient treatment of those dependencies. The implementation of simple island constraints is also given to emphasize the relevance of the difference-list technique. Finally, the problem section addresses grammar extensions such as noun complements and modifiers. This chapter is the central part of the book and provides a good basis for the reader who is willing to write his own grammar. However, I feel that the semantic-interpretation section should have been a little more developed, since 1. the main goal of language processing is to understand a sentence or a text via its semantic representation, and 2. the semantic interpretation of other constructions like modifiers requires improvements of the basic construction method presented here and would be worth presenting at this level. The probable reason for this lack of development is that relatively few works have been undertaken in the area of semantic representation and interpretation using Prolog.

Chapter 5 introduces metalogical facilities offered by standard Prolog (e.g., built-in predicates like *call*, *cut*, *set of*) and then proposes a simple dialogue program. Chapter 6 is entirely devoted to the construction of interpreters for DCGs. Interpreters mainly permit one to circumvent computational problems like left-recursion in grammar rules, implementing parsing strategies

different from the strategy imposed by Prolog, and partly automating the writing of rules. For example, arguments having a regular and predictable structure (parse tree building, logical form construction, etc.) can be automatically included in grammar rules. Such interpreters are written in Prolog and they transform a Prolog-like program into one that is directly executable. Fundamental techniques are proposed and illustrated for DCGs, such as automated construction of proof trees, interpretation of filler-gap dependencies, partial execution, and tabular parsing (i.e., chart parsing). The book ends with annotated sample programs that integrate most of the material presented in the preceding chapters. The bibliography is relevant and comprehensive, given the aims of the book.

To conclude, I feel that this book includes most of the material one would wish to see in an introductory book. By a small number of linguistic concepts and constructions, the authors present the main techniques a grammar writer can use in a large number of situations. There are, however, some points I would have liked to see included in this book. Very few parsers directly use the linguistic framework proposed here but rather are inspired by GPSGs, LFGs, HPSGs, UGs, etc. Thus it would have been of particular interest to have guidelines on Prolog techniques useful to such computational-linguistics frameworks. Some aspects, like feature percolation and control in GPSGs, are probably not trivial to deal with.

In spite of these nonessential restrictions, I feel that this book should be read by everyone in the field having a minimum interest in Prolog. This very convincing book also suggests the reading of more complex publications in the field like Dahl and St.-Dizier (1985, 1988) and JLP (1986).

References

Colmerauer, A. 1978. Metamorphosis Grammars. In *Lecture Notes in Computer Science*, 63: 133–189, L. Bolc, (ed.). Springer-Verlag.

Dahl, V. and Saint-Dizier, P. (Eds.) 1985, 1988. *Natural Language Understanding and Logic Programming I* and *II*. North-Holland.

Sterling, L. and Shapiro, E. 1986. *The Art of Prolog*. MIT Press, Cambridge, MA.

JLP 1986. Special Issue of the *Journal of Logic Programming* on Natural Language Understanding 3(4), December 1986.

Patrick Saint-Dizier was one of the organizers of the Workshop on Natural-Language Understanding and Logic Programming, and is editor of the forthcoming proceedings, to be published by North-Holland. In 1987, he was a visitor at Simon Fraser University. Saint-Dizier's address is: Institut de recherche en informatique et systèmes aléatoires, Campus Universitaire de Beaulieu, Avenue du Général Leclerc, 35042—Rennes Cédex, France.

Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems (The Kluwer International Series in Engineering and Computer Science: Natural-Language Processing and Machine Translation)

Masaru Tomita

(Carnegie-Mellon University, Pittsburgh, PA)

Boston, MA: Kluwer Academic Publishers, 1986, xviii + 201 pp.
ISBN 0-89838-202-5, $39.95 (hb)

Reviewed by
Mirosław Bańko
Warsaw University

This book introduces a context-free parsing algorithm that is intended to be particularly useful in natural-language processing. Practical applications of the algorithm are discussed, including on-line parsing, interactive machine translation, and a technique to disambiguate a sentence by consulting the user interactively. The book opens with a diagnosis of the current state of work on parsing context-free languages. Two kinds of algorithms for context-free languages are available: specific algorithms for programming languages, and general algorithms intended to handle any context-free grammar. Neither of them, however, is suitable for natural languages. The former are very fast but not powerful enough to cope with some syntactic phenomena of natural languages; the latter are even too powerful, and therefore not so efficient as they could be. Tomita's solution to the problem is to find something in between: an algorithm that would be nongeneral in the class of context-free languages yet still powerful enough to handle the great many ambiguities inherent in any natural language.

The algorithm described can be viewed as an extension of the LR parsing algorithm, which is used for programming languages (Aho and Ullman 1977). Like the LR algorithm, it works strictly from left to right and is entirely table-driven; unlike the LR algorithm, however, it is intended to parse ambiguous sentences, and does it most efficiently, using the data structures that have been specially designed for it. The current state of parsing is represented in the form of a directed acyclic graph, and no part of an input sentence is parsed more than once in the same way. All the possible parses of an ambiguous sentence are produced in parallel and stored in a **packed shared forest** (PSF) for later computation. While the number of parses grows exponentially, the size of the PSF increases polynomially. The only other algorithm that produces a polynomial-sized parse forest without requiring a grammar to be in Chomsky Normal Form is Earley's (1970) general algorithm intended to handle any type of context-free grammar. Tomita shows, however, that Earley's original algorithm has a