Natural Language Processing for Prolog Programmers

Michael A. Covington
(University of Georgia)

Englewood Cliffs, NJ: Prentice-Hall,
1994, xvi+348 pp; hardbound, ISBN
0-13-629213-5, no price listed

Reviewed by
Ken Barker and Stan Szpakowicz
University of Ottawa

The title says it all: it is truly a textbook for Prolog programmers, for better *and* for worse. It presents simple NLP for experienced Prolog programmers.

The book is written in a casual (occasionally *quite* casual) style and is eminently readable if you are reasonably comfortable with Prolog. The presentation has a satisfying flow. The book feels well planned, and is certainly well executed, considering its goals stated in the preface. Editing is admirably meticulous, formatting nearly flawless. The many examples in Prolog are clear and work well. Prolog code for examples is available by *ftp*. References are abundant and useful.

The book has nine chapters and two appendices:

1. Natural language
2. Templates and keywords
3. Definite-clause grammars
4. English phrase structure
5. Unification-based grammar
6. Parsing algorithms
7. Semantics, logic, and model theory
8. Further topics in semantics
9. Morphology and the lexicon

A. Review of Prolog

B. String input and tokenization

Chapter 1 offers a definition of NL and NLP, and a review of the usual linguistic strata. It also advocates Prolog as an NLP tool. Chapter 2 offers a pattern-matching approach to NL interfaces to operating systems and databases. Chapter 3 introduces parse trees, phrase-structure rules, and Prolog's DCG notation; the chapter also hints at parsing and semantic analysis (treated separately in Chapters 6, 7, and 8). Chapter 4 develops a set of phrase-structure rules for the main English parts of speech; it also goes beyond phrase-structure rules and gives a simple presentation of ID/LP rules and transformational grammar. Chapter 5 introduces a unification-based grammar formalism and an extension to Prolog for this. Chapter 6 describes and compares top-down, bottom-up, left-corner, and chart parsing as well as Earley's algorithm. Chapter 7 offers techniques for representing, in Prolog, knowledge from English sentences. Chap-

ter 8 discusses language translation, word-sense disambiguation, and understanding events. Chapter 9 presents some background on morphology as well as some models and techniques for computational morphology. Appendix A is a thorough review of Prolog (it accounts for 10% of the book!). Appendix B provides Prolog code for string input and tokenization, used elsewhere in the text.

This is not the first book that deals with NLP and Prolog. Gazdar and Mellish (1989) wrote a book primarily on NLP, which uses Prolog to illustrate linguistic concepts. The programming language is entirely secondary—there are parallel versions of the same textbook that use Lisp and Pop-11. Pereira and Shieber (1987) proposed to do subtle, advanced NLP in Prolog. Although their linguistic decisions might not appeal to everybody, they are never shallow. Pereira and Shieber's approach was the kind of introduction to NLP and Prolog that we find appealing to people seriously interested in computational linguistics, who need a software tool.

Covington's book can be perhaps characterized as a Prolog book that concentrates on NL-related issues. Readers familiar with his co-authored *Prolog Programming in Depth* (Covington, Nute, and Vellino 1988) will appreciate the similarities between its Chapter 13 ("Natural language processing") and the book under review. The latter may even be thought of as an extension of the former.

Covington accurately describes the present book (p. xv) as "first and foremost a book of *techniques* .... It is not a comprehensive handbook of NLP." However, while it indeed is not a comprehensive handbook, it does attempt to cover or at least mention in passing an astounding number of NLP issues and problems. Unavoidably, many of them get a rather skin-deep treatment. One problem with such an approach is that a book of techniques without depth risks coming across as a cookbook. And although cookbooks often have good recipes, they don't necessarily teach you much about good cooking.

Here's an example of a technique overshadowing theory. In the book there is an implementation of lambda calculus for compositional semantics. The implementation, however, is merely a neat Prolog device that gets Prolog to unify the variable arguments of structures with different functors. After the briefest explanation (section 2.3.6) "*lambda*" becomes merely a name for a specific form of Prolog terms. Not a word on *why* lambda calculus can represent certain linguistic phenomena, nor on *what* those phenomena are.

The Prolog techniques for NLP are presented in a fairly consistent format:

- here is an NLP problem;
- here is the author's simple solution in Prolog;
- here are exercises for the reader to improve on the solution.

This is not, perhaps, a format that stimulates deeper understanding. Solutions thrust upon the reader risk being memorized without due generalization.

The exercises are heavily biased towards Prolog. Too many of them are actually Prolog problems with little to do with linguistics. Some of them ask the reader to figure out why some piece of Prolog code works. This usually requires a good knowledge of Prolog and maybe even Prolog implementations. Nothing to it, if you *are* a Prolog programmer (fair enough; just recall the book's title) and you are willing to buy off-the-shelf components and refrain from delving into the underlying theories and algorithms.

Numerous exercises call for maintaining or enlarging the author's ready-made solutions. The reader is not really expected to come up with her own ideas: there is

no challenge to think linguistically, though there is challenge to think Prologically.

A fresh convert to NLP may find that the NL issues discussed in the book often appear to be very easily solved. For example, section 8.2 introduces machine translation, produces a solution, and gives exercises, in the space of four pages. The treatment looks so enticingly simple that the author himself found it necessary to have an extra page at the end (section 8.2.5) explaining "Why translation is hard". The same goes, *mutatis mutandis*, for section 8.3 ("Word sense disambiguation") and section 8.4 ("Understanding events"). It is small consolation that the author tells us (p. 233) this chapter will be "a whirlwind tour." It is not. It may be a bird's-eye view, but the bird is no falcon.

In a way, the very same concern holds for the whole book. The author says (p. xv): "It is meant for computer science students and working programmers who know Prolog reasonably well but have little or no background in linguistics." We are not sure whether they will come away from this book with much more than they arrived with. They may *think* they do, lured by the seeming facility of all those elegant, succinct Prolog morsels. However, unless additional material is covered in a course that uses this textbook, the reader will have only seen many short and sweet programs and many learned technical terms.

The book should be of good use in a specialized undergraduate course, and could also serve as a dependable technical supplement in a linguistically weightier graduate course for students in computing. Apart from such students, the book may appeal to a very limited audience: people who know Prolog well and may need some NLP for their mainstream programming or applications. Linguists might be overwhelmed by the technicality, and if they got past the technicality, the linguistic content remaining after stripping away the programming layer might not turn out to be too edifying for specialists. Perhaps they would appreciate Chapter 9 with its clear and appealing presentation. On the other hand, they might miss a serious mention of some *real* NLP systems.

Covington asks us (p. xv) to "judge this book by what it contains, not by what it leaves out." What it leaves out is deep linguistic motivations. What it contains is some clever and useful Prolog solutions to simplified NLP problems. So if that's what you need, this book is for you.

References

Covington, Michael; Nute, Donald; and Vellino, André. (1988). *Prolog Programming in Depth*. Glenview, IL: Scott, Foresman.

Gazdar, Gerald and Mellish, C. S. (1989). *Natural Language Processing in Prolog: An Introduction to Computational Linguistics*. Wokingham: Addison-Wesley.

Pereira, Fernando C. N. and Shieber, Stuart, M. (1987). *Prolog and Natural-language Analysis*. Stanford: Center for the Study of Language and Information.

Ken Barker is a doctoral student in computational linguistics at the University of Ottawa. His thesis deals with building semantic networks from parse trees in Prolog in the absence of precoded semantic knowledge. He has published one journal paper and three conference papers. *Stan Szpakowicz* is a professor of Computer Science at the University of Ottawa. His research interests include text analysis for knowledge acquisition. He has published about 80 refereed papers and four books. The authors' address is: Department of Computer Science, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5; e-mail: kbarker@csi.uottawa.ca and szpak@csi.uottawa.ca.