processing. *LDOCE* has some attractive special features, but so do other dictionaries. There can be little doubt, on the other hand, about the importance and value of the kind of research reported in this book.

References

Hofland, Knut and Johansson, Stig. 1982 *Word Frequencies in British and American English.* Norwegian Computing Centre for the Humanities, Bergen.

Jansen, J.; Mergeai, J.P.; and Vanandroye, J. 1987 Controlling *LDOCE*'s Controlled Vocabulary. In: Cowie, A.P., ed., *The Dictionary and the Language Learner* (Lexicographica, series maior, 17). Niemeyer, Tübingen, 78–94.

Mitton, Roger. 1986 A partial dictionary of English in Computer-Usable Form. *Literary and Linguistic Computing* 1:214–215.

Sampson, G.R. 1989 How Fully Does a Machine-Usable Dictionary Cover English Text? *Literary and Linguistic Computing.* 4:29–35.

Walker, D.E. and Amsler, R.A. 1986 The Use of Machine-Readable Dictionaries in Sublanguage Analysis. In: Grishman, Ralph and Kittredge, Richard, eds., *Analyzing Language in Restricted Domains: Sublanguage Description and Processing.* Lawrence Erlbaum, Hillsdale, NJ: 69–83.

Geoffrey Sampson is Professor of Linguistics and Director of the Centre for Computer Analysis of Language and Speech at the University of Leeds, Britain's largest unitary university. His computational linguistics research is corpus-based, and includes development of a system of robust parsing by stochastic optimization (Project APRIL). With Roger Garside and Geoffrey Leech he coedited *The Computational Analysis of English* (Longman, 1987). Sampson's address is: Department of Linguistics and Phonetics, University of Leeds, Leeds LS2 9JT, U.K.

Natural Language Processing in LISP: An Introduction to Computational Linguistics

Natural Language Processing in POP-11: An Introduction to Computational Linguistics

Natural Language Processing in PROLOG: An Introduction to Computational Linguistics

Gerald Gazdar and Chris Mellish
(University of Sussex and University of Edinburgh, respectively)

Workingham, England: Addison-Wesley, 1989
Lisp volume: xv + 524 pp.
Hardbound, ISBN 0-201-17825-7, £17.95
Pop-11 volume: xv + 524 pp.
Hardbound, ISBN 0-201-17448-0, £17.95
Prolog volume: xv + 504 pp.
Hardbound, ISBN 0-201-18053-7, £17.95

Reviewed by
Kwee TjoeLiong
University of Amsterdam

This is a very interesting and intriguing array of textbooks to read, to compare, and to review. It also has been a rather hard job for me to do so. The last paragraphs try to explain why. First of all, however, an objective and factual summary of the contents and form of Gazdar and Mellish's *NLP in X: An Introduction to CL,* where *X* is instantiated to one of {*PROLOG, POP-11, LISP*} (reviewer's shorthand).

Quotations can be helpful as the shortest way to give you a rapid impression. From the letter that the book review editor sent me is this encouraging line: "They are really three separate versions of the same book, so there's not nearly as much reading as there first appears." Therefore, one of the titles is treated here as prototypical (to wit, the Prolog volume). Whenever they differ, the other two are, subjectively, considered as derivative.

I am going to quote amply from the authors' Preface, since it is a characterization of the book in their own words. It is neatly split into sections, and I distinguish three aspects: 'What,' 'What Exactly,' and 'In What Way,' each aspect being handled in a pair of consecutive sections of the preface.

What: From the first two sections, *Audience* and *Coverage:*

> This book is aimed at computer scientists and linguists at undergraduate, postgraduate or faculty level, who have taken, or are concurrently taking, a programming course in *X*. . . . The book is specifically intended to teach NLP and computational linguistics: it does not attempt to teach programming or computer science to linguists, or to provide more than an implicit introduction to linguistics for computer scientists. . .
>
> The major focus of this book, as of the field to which it provides an introduction, is on the processing of the orthographic forms of natural language utterances and text. [No issues in speech, because those are] topics that deserve books to themselves, books that we would not be competent to write. Most of the book deals with the parsing and understanding of natural language, much less on the production of it. This bias reflects the present shape of the field, and of the state of knowledge. . .
>
> The book is formally oriented and technical in character, and organized, for the most part, around formal techniques. The perspective adopted is that of computer science, not cognitive science. . . . We concentrate on areas that are beginning to be well understood, and for which standard techniques . . . have begun to emerge. . . . [Hence,] a good deal more time on syntactic processing than on semantic or pragmatic processing. . . . Discussion of developments at the leading edge of NLP research, on such topics as parallel parsing algorithms, the new style categorial grammars, connectionist approaches or the emerging implementations of situation semantics and discourse representation theory are excluded altogether or relegated to the further reading sections. . . . A less readily excusable omission is any consideration of the role of probabilistic techniques in NLP. [But . . .]

probabilistic NLP work is largely restricted, at present, to those few centres that have the necessary data and the resources to process it.

What Exactly: Now that they have roughly sketched what the book is about and what not, it is possible for the authors to specify some more details. In the fourth section, *Contents,* the table of contents, that is, the list of chapter and section titles, is paraphrased in running text. I in turn squeeze it back into an enumeration of key words a bit further on below. In the third section, *Grammar formalism,* it is announced that:

> Rather than attempt to survey some subset of [the many different grammar formalisms adopted by computational linguists, over the years and even today,] . . . we have simply adopted one, PATR. . . . PATR is relatively widely used in the NLP community. . . . From the perspective of this book, it represents a very clean, uncluttered, basic unification grammar formalism that is sufficiently expressive that pretty much anything one might want to say in another grammar formalism can be readily translated into a PATR equivalent. It is also rather straightforward to implement an interpreter for PATR in *X*, and we show how to do this.

In What Way: In conclusion, the authors describe their own way of handling the matter. From the fifth and sixth sections of the preface, *Organization* and *Programming language:*

> Each chapter contains material of two types. The first type is a relatively self-contained treatment of some theoretical topic. Here, computational issues are discussed without reference to the details of *X* or any other programming language. A person without much computational background should be able to get a feel for the issues in NLP by just reading this material. The second type contains extracts of programs, notes on techniques and exercises in *X*. . . . We have attempted to grade exercises as follows . . . [easy, intermediate, hard, project].
>
> In our view, for a computational linguistics or NLP textbook to be maximally useful, it must commit itself to a particular choice of real programming language. . . . The choice . . . has to recognize the major regional language divisions in computing and AI.
>
> We feel that the study of computational linguistics is only brought to life by actually writing and running programs. So, a textbook on the subject should provide a source of ideas and examples to stimulate the student's initial programming activity. It is much easier to follow a particular computational concept or algorithm, especially to those lacking a thorough computer science background, if it is expressed in a familiar programming language than if it is expressed (formally or informally) in some unfamiliar way. . . . We have chosen to base our discussions of programming and algorithms around examples in an actual programming language, namely *X*.

Let me now enter into the form and the contents proper. Here are some general statistics: the body of the text consists of slightly more than 400 pages; the remaining 100–120 pages are mainly taken by code listings (Prolog, 66 pages, and Pop-11 and Lisp, 80 pages, for 54 listings; not always the same examples in all versions) and a bibliography (some 20 pages for some 350–400 titles); there are solutions (mostly hints) to (a few) selected exercises, a name index, and a general index. Each of the ten chapters is subdivided into short sections (with an average of eight or nine per chapter). The chapter lengths cluster around an average of 40 pages, with, exceptionally, 20 pages for Chapter 1, and 65 pages (plus or minus 3 for the different language versions) for Chapter 7 (which, for its contents, might as well have figured as two, or one-and-a-half, separate chapters). Altogether the chapters can be grouped into three chunks, in the proportion of 100:175:125 pages, as follows (for each part I have supplied an appropriate characteristic motto):

Part 1, the first three chapters ("What happened before"): One chapter on introductory generalities and two on the classic stock of automata, transducers, and transition network techniques (finite-state; pushdown or recursive; augmented).

Part 2, the middle four chapters ("Today's frontiers"): Basic concepts are introduced and illustrated in an up-to-date, feature-theoretic, unification-based directed-acyclic-graph–oriented setting: grammar in Chapter 4 (rules, structures, context-freeness); parsing, search, and ambiguity in Chapter 5 (bottom-up, top-down; breadth-first, depth-first; determinism, look-ahead); charts in Chapter 6. The culmination point is reached in Chapter 7 on feature theory (graphs, subsumption, unification), where almost everything is being resumed and connected to almost everything else, the lexicon included.

Part 3, the last three chapters ("The future has already begun"): A selection of established and recent topics, from semantics in Chapter 8 (meaning representation language, which may remind one of logical form and of knowledge representation), via its corollaries in Chapter 9, question-answering (database query), inference (backwards, forwards; frames), and primitives (inheritance, defaults; semantic networks), to pragmatics in Chapter 10 (the variety of different roles of noun phrases; prediction and scripts; discourse structure).

The Prolog version is distinguished from the other two in that is has practically nothing on ATNs (which means twelve fewer pages in Chapter 3), and no random generation either (six fewer pages in Chapter 7). This is made up for by the insertion of definite clause grammar here and there in the remaining middle chapters.

The central part of the Prolog version is, obviously, the

pièce de résistance of the work. Why do I think it is? Because it is here that the authors show their commitment. And also because it might be possible to use this part as a classroom text independently from the other two. I would dare to bet that Part 1 is not really a prerequisite for it. Part 3, on the other hand, is not really an essential or natural sequel to it. In turn, Part 3 possibly does not require Part 2 either, except for the fact that a predicate logic formula can be represented as, or translated into, a directed acyclic graph. Furthermore, there is far less programming in Part 3. In short, Part 3 functions rather as a first introduction, albeit in a feature theoretic coating, to issues in formal semantics, philosophy of language, and discourse analysis.[1]

Some authors of classroom texts want their book to be self-contained. Gazdar and Mellish do not have this desire; witness the quotation above from their Preface. Indeed, linguists who have no knowledge of programming and who missed the explicit clues given there will soon be at a loss and give up, or else skip the programming part and try "to get a feel for the issues in NLP" anyhow. Now, imagine computer scientists who have little or no knowledge of what linguistics is and who missed the implicit clues given in the Preface (they will also miss the warning that is to follow here, since they are not among the readership of this journal anyway). Chances are, they will be able nevertheless to finish the book. On the one hand, that is certainly a merit of the book; on the other hand, is it not also because the story it tells is a little bit misleading? At least, the story is a bit risky. Grammars and linguistic examples are invariably of the toy kind. Interesting complex structures are hardly shown. One gets a strong feeling that the book is addressed to linguists in the first place, in order to persuade them into CL (an endeavor I support, for that matter), and that the main interesting problems in grammar are assumed to be known to the readers. But these problems are not treated at all, at most hinted at in some exercises. Why? I suppose, because they would need a lot more technique, being much more intricate than the toy examples. The necessary techniques, however, not only in the specific programming language but in general in the domain of concepts and formalisms, as they already possess a certain level of intricacy by themselves, must be presented and illustrated with the help of simple cases—an approach I approve of, with one proviso: students should realize that this is not the whole story. Are nonlinguists in general really aware of more interesting problems at an advanced level of grammar? I often doubt it. It is alarming how false an impression of NLP/CL people get from the lecture of just one chapter in an introductory book on AI: only simple examples are handled, and it is tacitly implied that the rest of grammar is pretty much alike. Knowledge of language is often mistaken for knowledge of linguistics. (Think of mechanical translation in the 1950s and early 1960s and the naive optimism it aroused at that time. What is, by the way, the first meaning of the word *linguist,* in most dictionaries and in daily life outside our field?) So, to use the jargon of movie rating, I have to give this warning: adult linguistic guidance is strongly advised (in the second dictionary sense of *linguist*).

Some users of classroom texts have been spoiled by a certain tradition of *Hints on How to Use This Book,* either as scheduling suggestions (during W weeks, M meetings per week, of H hours each) or in the form of a chart-like diagram (or a DAG—why not?) of the dependencies between the chapters. To the slight annoyance of those people, there are no such suggestions here whatsoever. Why? Is it forgetfulness? Or is it because the chapters are equally important and one simply has to work through them one by one in linear order? Is it indulgent liberality, the wish to avoid imposing strict schedules upon others? Or is it a challenge, in the sense that one should find out oneself which chapters to treat in full, which sections to skip if time presses, and in which pace? Yet some guidelines in one way or another would probably have been appreciated, for instance, by those with little experience in teaching this material, possibly in order to avoid overcharging their class, or to check that their demands were reasonable after all.

If no other points are mentioned here that might be considered doubtful or mistaken, that does not mean that I entirely agree with every detail I set my eyes upon. For instance, minor errors have, inevitably, arisen in the wealth unfolded in the recommendable further reading sections and in the extensive bibliography. It is left to attentive readers to find those corrigenda et addenda by themselves. I pass over them in silence.

Literature in NLP/CL, as in all new emerging fields, has for a long time been suffering from a "case history" syndrome. Textbooks, or overview chapters in AI books, were typically surveys of the various research projects, telling over and over again the stories of Eliza, SHRDLU, Lunar, and so on. It was often unclear what the field consisted of exactly; even its name was problematic.[2] In the title of the work under review, the authors have taken a stand: NLP is technology, and they present it as a road toward CL. CL is much more than just (syntactic) NLP. CL is theory, and it is based not only on grammar but also on formal logic and on discourse analysis. The authors have demonstrated this point of view in their choice of one single particular notation throughout, in the coherence and consistency of their treatment, and in the extent of their subject matter.[3]

To sum up, then, this is my evaluation of the quality and importance of the work at hand. In my opinion, this book is in general fairly good, and in part very good. It is important enough for the interested and well-prepared reader (who took a full-year linguistic course, and has done at least a moderate amount of programming) to try to get a copy in her or his own preferred language version.

As a rule, this kind of text has two types of readership. I don't mean those who approach the field from computer science and those who approach from linguistics; on that aspect I have already made an observation. Now, I have another distinction in mind: between what I call Readers-from-Outside and Readers-from-Inside. People from both

categories are interested in the question *What can you do with natural language on a computer?* The first group take *you* in the generic reading. They would like to know, to see, to get acquainted with, what one can do, what people can do. The second group gives *you* a specific interpretation, because they want to find out, to study, to learn, what the addressee can do, that is, what *they themselves* can do. By association, I count among the last category not only students but also those who teach them, that is, those who put the question to the addressees.

In reading and reviewing a new classroom text on NLP/CL, it is difficult not to compare it to other books one is already familiar with—the more so, when one such other book is generally known, widely used, and frequently referred to: Winograd (1983) (see Cohen 1986).[4]

It is here that the biased part of the review starts. I am using Winograd's book in my classes and I am happy with it. I admit the merits of the work under review, and won't take back anything I have said above. I perfectly realize that the next few paragraphs express a one-sided view that not everybody would share, or would agree with; nobody has to. It is one particular approach, heavily influenced by one particular situation. The typical audience in that situation has not quite taken a full-year linguistic course, and has done only a very modest (less than moderate) amount of programming.[5]

In an introductory NLP/CL course, I claim—especially in an NLP/CL *cum* programming course—one should be made aware of three distinct levels of reflection, or three subsequent stages. At the linguistic level, a clear definition and description is needed of the phenomena to be captured. At the algorithmic level, the data and the goals have to be set, and a method has to be traced out. At the coding level, the formal algorithm must be translated into code of a real programming language. The last two things must absolutely be kept apart.

These three levels are well distinguished in Winograd's book. Language and linguistics are treated in two chapters (on transformational grammar, and feature and function grammars) and two appendices (syntax of English and recent developments in TG), together good for about forty percent of the book. A drawback is that theoretical approaches become obsolete. But interesting complex phenomena and structures are here to stay. Algorithms are carefully presented in a clear formalism (which, unfamiliar as it may be at first, is not difficult to understand) and are kept visually well separated, displayed in special boxes. One big advantage, for my goals, is that the coding level is simply left out, that is, the book is programming language–free. Thus it is suitable for giving students the training they need in writing programs. At a certain level, coding is a creative activity. Moreover, coding a given algorithm has a twofold effect. It invites students to reflect, both upon the procedures they thought to have understood, and upon the expressiveness of the language they have to acquire fluency in. Winograd's Chapter 2 (patterns, FTNs), Chapter 3 (context-free parsing) and Chapter 5 (RTNs, ATNs) are very useful for this goal and for our audience (as a matter of fact, I have been myself preparing a little *Prolog Companion to Winograd's LCP1: Syntax*). The same subjects can be found in Gazdar and Mellish's Chapter 2, Chapters 4 to 6, and Chapter 3, respectively.[6]

Some people object that, in this way, students have to re-invent the wheel, an activity they consider unnecessary. I contend that before you have your students tackle new and big (and may be as yet unsolved) problems, you'd better let them have some experience in tackling small and simple problems, the solutions of which you know and they don't. In this way you will be able to help them and to coach them, and you give them the opportunity to grind their knives at their own levels. The work under review lends itself better to another strategy. Here, the programming language is an essential tool for exposition: algorithms are often presented in the form of a working program. Thus, a twofold goal of another type is pursued. Not only do students learn to read and understand real code, but they are also given, for free, the techniques that may lead them smoothly into the advanced realms of research. Winograd's book and this work are each better suited for one of two rather opposite pedagogical styles. Both styles have their supporters. To be able to profit maximally from the study of this work, I think, students need a higher level at least in programming than used to be required for our introductory NLP/CL classes. This book as a classroom text makes it harder for relatively inexperienced programmers to develop (and to show) their creativity in writing code. Regrettably so, for re-inventing the wheel can be a satisfying and rewarding activity, worthwhile of investment.

Notes

1. And why in particular the Prolog version? Because I think this version is their prototype text. Here is some circumstantial evidence. As most readers will remember, the second author is also co-author of a well-known book on Prolog. The authors already used an early draft of the Prolog version when teaching their CL course at the LSA Linguistic Institute, Stanford 1987. The formalism they adopt has an inherently declarative character. The font they use for sample conceptual objects (such as grammar rules and feature structures) is different from the font they use for program code, but logic formulae all over the three editions are typeset in the same conspicuous boldface as the Prolog code, widely distinct from the Pop-11 and Lisp lightface sans-serif fonts. The bibliographies have about 350 titles in common, to which a varying number of language-specific references is added in the different editions: 8 for Pop-11, 20 for Lisp, and not fewer than 49 for Prolog. The preface in the Prolog version bears the date of February 1989, and, although the Pop-11 preface shows the same month, the Lisp version (which in the noncode parts is quite identical to the Pop-11 one) has its preface dated March 1989.

   Post scriptum: I would like to add that this conclusion, although an easy one to come to, turned out to be completely wrong. In fact, Mellish informs me: "The Prolog-specific material was created relatively late in the writing, and was not parasitic on the others, or conversely. Actually we had a quite elaborate . . . set-up that allowed us to write all three books simultaneously . . . without any text having a priority."

2. Compare the rather amusing confusion produced by the most prominent bibliography of the field published to date, Gazdar *et al.* (1987). The book is called *NLP in the 1980s* on the cover and title page, for The Outside World, so to speak. Inside the book, the list itself is called "CL in the 1980s," for The Inside World. In addition, it is computer-accessible as *CLBIB* at Stanford, and letter mail must be sent to *CLBIB* at the School of Cognitive Sciences, University of Sussex.
3. It is to be hoped that this tendency will continue to develop, similar to the line of growing insight shown in the sequence of (titles of) books by Nilsson: *Problem-Solving Methods in AI* (1971), *Principles of AI* (1980/1982), and *Logical Foundations of AI* (1987, with Genesereth).
4. In this survey of CL courses Winograd (1983) as the most frequently cited reference, with 23 citations in 51 courses described as 'only CL'; the most frequently cited references in all 76 courses (including 25 described as 'courses with topics other than CL')—46 courses within North America and 30 outside North America—are: Winograd 30, King 10, Tennant 8, Schank and Riesbeck 7 (Cohen 1986:4).
5. A quick sketch of the local context at our department: University entrance level in Europe is in general considered to be equivalent to American junior college graduation. Four years at a European university are comparable to junior and senior year plus graduate studies up to Master's thesis in the U.S. Students enter our seven-to-eight-quarter CL program after at least one year study at a language department, and they have to take (or to have taken) optionals for two additional quarters. The core program in the first two to three terms consists of classes in linguistics, formal logic, and computer science (that is, introductory programming and an introduction to formal languages and automata theory). Students used to take the Winograd course at some point around the end of the core program and the beginning of the advanced terms. We are currently engaged in incorporating it into the core program (as of Spring 1990).
6. Setting aside the fact that ATNs are admittedly in decline, it is surprising to see how the choice of a language can determine which subjects to treat: see my previous remark on the differences between the Prolog edition and the other two versions. Programming an ATN in Prolog is not so thoroughly perverse as Gazdar and Mellish think it is (p. 96), nor is it too difficult. Some of our students succeed in doing so without much contriving, and it looks like really nice Prolog.

References

Cohen, Robin (compiler). 1986 Survey of Computational Linguistics Courses. *Computational Linguistics*. 12:course survey supplement.

Gazdar, Gerald; Franz, Alex; Osborne, Karen; and Evans, Roger. 1987 *Natural Language Processing in the 1980s: A Bibliography*. Center for the Study of Language and Information, Stanford, CA.

Genesereth, Michael R., and Nilsson, Nils J. 1987 *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA.

Nilsson, Nils J. 1971 *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, New York, NY.

Nilsson, Nils J. 1980 *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA. Also: 1982 Springer, Heidelberg.

Winograd, Terry. 1983 *Language as a Cognitive Process. Volume 1: Syntax*. Addison-Wesley, Reading, MA.

Kwee TjoeLiong (in the U.S.: *TjoeLiong Kwee*) is a mathematician and a lecturer at the Department of Computational Linguistics, University of Amsterdam, is teaching courses in programming and in computational linguistics, and is doing research on generation and on modeling and testing of linguistic theories. Kwee's address is: Department of Computational Linguistics, University of Amsterdam, Spuistraat 134, 1012 VB Amsterdam. The Netherlands. E-mail: tl@alf.let.uva.nl

---

Modelisation du dialogue: Représentation de l'inference argumentative (Modeling Dialogue: Representation of Inferential Argumentation)

Jacques Moeschler
(Université de Genève)

Paris: Hermès, 1989, 266 p. (Langue, raisonnement, calcul)
Hardbound, ISBN 2-86601-191-0, FF 200

Reviewed by
Klaus Schubert
BSO/Research

The other day a professor of linguistics told me about one of his student's works:"It is very mathematical but intelligent." If you want to know what this little *but* means, implies, and reveals, read this book.

Jacques Moeschler, who is engaged in the linguistics of French at the University of Geneva, offers a model for recognizing and formally representing the structure of dialogues. His work is a contribution to the insufficiently explored field of discourse analysis in the original sense of the term, that is, the analysis of oral conversation. Moeschler's analyses are based on a corpus of French telephone conservations and similar materials, but many of his findings may apply even to the broader field of text linguistics.

A special focus in Moeschler's argumentation is on the ways that propositions are linked up to form a text or a dialogue. Many text grammarians distinguish various kinds of linkedness in dialogues (cf. de Beaugrande and Dressler's *Kohärenz* vs. *Kohäsion,* 1981: 3ff). Moeschler's terms are *pertinence* for the connectedness in the context and *coherence* for the connectedness in the discourse situation. The central element in Moeschler's reasoning, which gives the book its special interest and also its individual flavour in the current flood of text-linguistic studies, is the notion of *connecteur.* To put it simply, such connectors are function words or word groups that link up propositions logically, temporally, or otherwise. Examples (which, due to their very nature, normally cannot easily be glossed out of context) are *et, ou, ne pas, quand même,* and *alors.*

The theoretical background of the work is to be sought in predicate logic and speech act theory. Among the major sources are works by Ducrot, Austin, and Grice. The book falls into three parts, of which the first is mainly dedicated to a predicate-logical analysis of the meaning of connectors. Although Moeschler pursues this analysis in much detail, the main object of his analysis remains the word itself, rather than some underlying semantic or logical representation. To my personal taste, this is the most important virtue of the book: Moeschler has understood that human language is richer than formal representations (cf. Schubert 1988a: 137–138). He gives this insight right in the introduction (p. 10), stating that in natural language, there are a large number of connectors that lack equiva-