PREFACE

> I have, notwithstanding discouragement, attempted a Dictionary of the English Language, which, while it was employed in the cultivation of every species of literature, has itself been hitherto neglected; suffered to spread, under the direction of chance, into wild exuberance; resigned to the tyranny of time and fashion; and exposed to the corruptions of ignorance, and caprices of innovation.
>
> When I took the first survey of my undertaking, I found our speech copious without order, and energetic without rule; wherever I turned my view, there was perplexity to be disentangled and confusion to be regulated; choice was to be made out of boundless variety, without any established principle of selection; adulterations were to be detected, without a settled test of purity; and modes of expression to be rejected or received, without the sufferages of any writers of classical reputation or acknowledged authority.
>
> Samuel Johnson, 1755/1938
> *Preface to the English Dictionary*

From the time he could pick up a crayon, Pablo Picasso had the ability to sketch pictures that accurately depicted the things he saw. He was exceptional. Most of us spent our early years following the dots, painting by numbers, and drawing on tracing paper held over some else's picture. If you are a Pablo Picasso of the computer, and if computational linguistics comes easy to you, this book may not be for you. But if you would like a follow–the–dots approach to natural language processing, linguistic theory, artificial intelligence, and expert systems, you will like this presentation. The basic idea is to present meaningful answers to significant problems involved in representing human language data on a computing machine. My main focus is on the grammatical devices underlying constructions of English, French, and German. I use the absolute minimum required from any computer hardware or from either of the computer languages discussed: Prolog and UNIX.

This book, which perhaps should be subtitled *start here*, offers a hands–on approach to anyone who wishes to gain a perspective on **natural language processing**, the computational analysis of human language data. All of the examples in this book are illustrated using computer programs that run. The optimal way for a person to get started is to run these existing programs to gain an understanding of how they work. After

gaining such familiarity, you can start to modify the programs and eventually start to write your own. The optimal way to learn to create programs is to operate on live ones by modifying existing code. All programs discussed, and a shareware version of Prolog to execute them, are available on a disk (see Appendix I) or from the New York University public domain bulletin board (see Appendix II). All programs run on free versions of Prolog–2 (IBM) and Open–Prolog (Apple). See *How to Use This Book* for a pictorial representation of the book's organization. If you have a computer, you have no excuse for not running these programs as you work through the book.

My main goal is to show the reader how to use the linguistic theory of Noam Chomsky, called **Universal Grammar**, to represent some of the grammatical processes of English, French, and German on a computer using the Prolog computer language. I assume the reader knows very little about Prolog and less about Chomsky's linguistic theories. The Introduction, *What is Computational Linguistics?*, presents an overview of the problems and problem areas I discuss.

Prolog, short for ***PRO****gramming in* ***LOG****ic*, differs from most computer languages in that the programmer usually does not need to tell the computer the procedural steps to follow in order to find an answer. A Prolog program consists of a database, relations among items in the database, and a set of boundary conditions that identify correct, or acceptable, answers. Although the operation of Prolog initially appears counterintuitive, and although the basic concepts underlying Prolog reflect the arcana of formal logic, Prolog is remarkably easy to learn for beginners if it is presented using simple examples and lots of diagrams. This introduction to Prolog offers simple examples and figures to illustrate most Prolog operations.

Chomsky's linguistic theories (generative grammar, universal grammar, principles and parameters, etc.) do not lend themselves to simplification because the theoretical significance of them derives from their logical "tightness." That is, from a small number of general principles that can be formalized in terms of logic and computer processes (hence, the relevance of Prolog in the exposition), it is possible to derive many of the seemingly exceptional and irregular properties of human language structure. Unfortunately, the diagrams offered to clarify ideas basic to sentence analysis in Chomsky's system often frighten people away.

Rather than plunge directly into the basic assumptions of Chomsky's system and attempt to derive the properties of human language by Cartesian deduction from the basic principles of Universal Grammar, I first discuss the types of problems that arise in Chomsky's system, and then turn to formal linguistic theories. My view of Chomsky's system has more in common with the methodological views of Gertrude Stein than of Descartes and Pascal.

Gertrude Stein wrote:

> When you are at school and learn grammar grammar is very exciting. I really do not know that anything has ever been more exciting than diagraming sentences. I suppose other things may be more exciting to others when they are at school but to me undoubtedly when I was at school the really completely exciting thing was diagraming sentences and that has been to me ever since the one thing that has been completely exciting and completely completing. I like the feeling the everlasting feeling of sentences as they diagram themselves. (Stein 1935: 81)

Following Stein's ideas, my goal is to use the computer language Prolog to diagram sentences using the types of diagrams that Chomsky has developed.

Chomsky and his group select human languages (English, Walpiri, etc.) and structures in human languages (questions, passives, relative clauses, etc.) because these languages and structures bear on the correctness or incorrectness of their linguistic descriptions and theories of mind. In selecting problems I depart from this strategy and follow the guidelines of Ms. Stein. Abandoning concerns of description and explanation, she focused on the interesting and ignored the boring. She claimed:

> Now in that diagraming of the sentences of course there are articles and prepositions and as I say there are nouns but nouns as I say even by definition are completely not interesting, the same thing is true of adjectives. Adjectives are not really and truly interesting... Beside the nouns and the adjectives there are verbs and adverbs. Verbs and adverbs are more interesting. In the first place they have one very nice quality and that is that they can be so mistaken. It is wonderful the number of mistakes a verb can make and that is equally true of its adverb... In that way any one can see that verbs and adverbs are more interesting than nouns and adjectives. (Stein 1935: 82)

Diagraming sentences in 1935 was fun for Ms. Stein when the diagrams fit on the back of an envelope and could be sketched with a pencil. But Ms. Stein's smile would soon yield to a frown with Chomsky– type diagrams. Even for a simple sentence like *Tess tried to go*, Chomsky's structural descriptions rapidly inflate – according to general principles, of course – to cover pages. This study shows how to program Chomsky's grammar into a computer using Prolog to diagram sentences using diagrams (structural descriptions) of the type developed by Chomsky's studies in theoretical linguistics. In terms of computer processes (derivations) and data structures (representations) I follow the theories of Noam Chomsky closely. In terms of problem selection and presentation, I follow the insights of Gertrude Stein.

I attempt to communicate Stein's enthusiasm for diagramming sentences by relegating to Prolog the boring job of drawing the giant Chomsky–type structural

representations. I focus on phenomena, such as ambiguity, that Stein felt enabled verbs and adverbs to make mistakes. But there are other sources of enthusiasm for this study. Stein said:

> One of the things that is a very interesting thing to know is how you are feeling inside you to the words that are coming out to be outside of you. Do you always have the same kind of feeling in relation to the sounds as the words come out of you or do you not. All this has so much to do with grammar and with poetry and with prose. (Stein 1935: 93)

Although not quite the way Chomsky would say it, Stein's point echoes a tenet underlying Chomsky's study.

According to Chomsky's **universal grammar**, human language structure directly reflects the structure of human intellectual capacities. The phonetic, phonological, morphological, lexical, syntactic, and semantic structures found in language reflect the pattern–recognition abilities, problem–solving skills, and memory–storage capacities of the human intellect. There are three main reasons for studying **linguistics**, the science of the formal structures of human languages and the computational mechanisms internal to the human brain that underlie those structures.

- Language is as much a part of any human culture as are religion, politics, and rituals of birth, marriage, and death.
- Human language in its outline and in its details has structures that can be formalized in terms of logic and mathematics and that are sufficiently rigid to support or refute formal theories that describe those structures. One of the main contributions made by Noam Chomsky to the development of linguistics as a science consists of his methodological perspective, which shows in detail how aspects of mathematical logic can be used to develop formal theories of human language structures.
- The human capacity to learn, know, and use a language reflects a genetically based species' specific biological endowment. In studying how language is acquired by children, linguistics focuses on the nature–nurture debate: To what extent is human knowledge part of the inborn structures of the human brain? What is the role of experience in learning? What data are necessary and sufficient to enable a child to acquire a human language?

My main hope is that I can communicate some of Gertrude Stein's excitement for diagraming sentences, even though the diagrams reflect an intricate Cartesian sort of deduction from Chomsky's tightly knotted formal theories of language. For the most part,

I focus on the language problems and the diagrams. I develop Prolog as a tool to handle the intricate deducing.

The diversity of the technological frontiers is one of the main hurdles facing anyone trying to get a handle on the linguistic and computational problems involved in getting a computer to learn, read, write, and perhaps understand sentences in English, French, and German. I have adopted the lifeboat approach as opposed to the steamship view. You would pack a lifeboat with only essentials, but you could stuff a steamship with things you might need on a rainy day. My approach shows the reader where to focus attention and concentrate efforts. I try to present only those terms that constitute benchmarks for defining problems and solutions. Some such selection is necessary for a reader trying to keep his or her head above the sea of linguistic terminology, computer jargon, and general technobabble that has emerged from some artificial intelligence, expert systems, and knowledge engineering approaches that try to implement human languages onto a computer.

In the text, any term in SMALL CAPITALS is a basic concept you will need to learn, hence, I define it in the text and illustrate it using a Prolog program that encodes some aspect of English grammar. Any term in *italics* is a technical term in the background literature that underlies this study and can be referenced in linguistics and computer science textbooks. It would be nice to learn, but you might postpone it until you have mastered more basic concepts. Technical terms that are neither italicized or small capitals you can learn if you want to, but they are not essential. Any term in **boldface** is a definition or the name of a program. The goal here is to use Noam Chomsky's UNIVERSAL GRAMMAR to develop a GENERATIVE GRAMMAR to describe issues in the *syntax, semantics*, and *morphology* of English, German, and French using the NONPROCEDURAL language Prolog.

All of the examples run in C–Prolog on a UNIX VAX. The syntax of C–Prolog is the same as DEC–10 Prolog. Almost all universities, colleges, and junior colleges have a machine capable of running C–Prolog. The programs also run on Quintus Prolog. If you have no access to a DEC–10 or a VAX, I show how you can obtain a shareware copy of Prolog–2 and Open–Prolog via E–mail from New York University's Public Domain Bulletin Board. With some (usually slight) modifications, all programs will run on an IBM PC or Macintosh in Public Domain Prolog. Some students have run the programs using Arity Prolog and Microlog Prolog. All of the examples follow the definitions, style, and punctuation in the book by Clocksin and Mellish (1981).

If you intend to run the programs in Prolog–2 on an IBM PC, you might refer to Dodd (1990). Dodd is the developer of Prolog–2 for the IBM. He stated in his preface: "Readers with access to an IBM PC or equivalent with 512K of memory or more are welcome to a free copy of the simplest version of Prolog–2, a smallish Prolog that will execute most of the examples in Dodd (1990). Write the author c/o Oxford University Press." (Dodd: vi) This simple version of Prolog–2 also runs most of the programs in this

book. Dodd's book is complementary to this one in that he provides a detailed analysis of the complexities of Prolog, but his book contains, as he stated, "no serious presentation of natural language systems." (p. vii)

The uninitiated person who wishes to gain hands–on experience with natural language processing, artificial intelligence, Chomsky's linguistic theories, and expert systems is beset with a cornucopia of materials. This book serves both as a prism and a lens between you and the huge stack of literature on these subjects. The book functions as a prism by breaking the basic ideas of natural language processing into the component parts and illustrating each of them using a simple example. The book is a lens in that it focuses on the most relevant and interesting parts of natural language processing and ignores the extremely important, but less interesting, questions concerned with the elegance, efficiency, speed, and machine–specific implementation of the solutions. If I can illustrate some point very clearly by having an understandable 2 page program, then I present the 2 page program even if the program could be replaced by a logically equivalent smaller program that is intellectually opaque because it presents the problem obliquely.

If any program could be made more understandable by changing it in some way, then the change was made even if it led to redundancy, inefficiency, wasted memory space, misuse of computer resources, excessive CPU time, or worse. As you gain an understanding of the operation of Prolog, you will see alternatives to the programs presented in this book.

Almost all of the examples encoded into Prolog come from current research in linguistics. When I focus on nonlinguistic data, it is to drive home a point. In Chapter 1 I discuss *intelligence* in several domains in order to clarify a basic issue. In Linguistics, the term *human intelligence* is understood as a species specific biological endowment that is common to all humans, just as binocular vision, binaural hearing, and other like properties are common to human beings. The concept of *human intelligence* that underlies IQ testing, which aims at exaggerating the differences among humans, is not the concept being explored. Chapter 5, in which I encode a bibliography into Prolog, serves as a flu shot against the two most common errors that frustrate beginning Prolog programmers. Most neophytes use capital letters when they are prohibited, and even worse, they forget to use the period. A bibliography has lots of capitals and punctuation and serves as an ideal training ground to learn the two most misused Prolog conventions: A capital is always a variable. All statements must end in a period, and a period anywhere but at the end of a statement is usually bad news.

The material in the *Foreword* derives from an interdisciplinary course funded by the New York University Humanities Council, *The Cultural History of Computers, Robots, and Artificial Intelligence.* I first offered the course to undergraduates in 1986. Given its favorable response, the Humanities Council funded research to develop an

interdisciplinary graduate course of the same name, offered in 1990. The course was cross–listed and offered for credit in several departments. Comparative literature students showed that Hamlet's reflective indecision was limited by the fact that he spoke English. Some art history students pointed out that it seemed to them natural for the exceptions and irregularities of English to be considered "simpler" than the regular forms. The ideas from this class led to the organization of this book and, in particular, to the order of the presentation of material.

The material in Chapter 1, *Natural Intelligence, Linguistics, and Prolog*, derives from an interdisciplinary faculty seminar, *The Information Society: Artificial Intelligence, Robotics, and Expert Systems*, funded by the Humanities Council from 1985 until the present. Each term we establish a schedule of speakers, one every two weeks. The speakers are from diverse fields in various universities; from IBM, Bell Labs, and various other corporate research facilities. Each speaker sends us materials that we read and discuss at a meeting the week before they speak. This guarantees each speaker a live audience that will understand his or her ideas. It was at these meetings that I was forced to develop answers to the questions: What is the *information society*? What is *artificial intelligence*? What is an *expert system*? What is *computer power*? Why study Prolog? Why study natural language computing in Prolog? How does research on generative grammar in the Linguistic theories of Noam Chomsky have any bearing on the productivity of workers in sector III, *the services sector*, of the economy? Insofar as this essay is eclectic, it derives its eclecticism from my attempts to clarify and justify my work to the heterogeneous array of friendly, but not necessarily sympathetic, participants at the faculty seminars.

The computational linguistics material in this book has been used in various Linguistics courses at New York University for three years. Although there may be some glitch that has snuck through, several generations of students have ironed out most of the kinks and wrinkles in the programs. I am quite certain that each program will execute precisely as indicated if you are using C–Prolog on a UNIX VAX. I am less certain about the possible outcome if you are running the programs in the shareware version of Prolog, which executes on an IBM PC. In this case, the results will depend on the properties of your specific IBM PC. For one term I had a dozen students testing the materials using the shareware version of Prolog on whatever computer they had at home. Most of the problems encountered were minor and easily remedied, even by a novice. The main difference between C–Prolog and the shareware Prolog is speed. When you run some programs on a 286 IBM PC, you will have lots of time to go get a drink of water, stretch your legs, or read the papers. Shareware Prolog on a 486 PC can sometimes outperform C–Prolog on the VAX.

Some of the terminology used in this book diverges from that one might find in a standard linguistics text or in a book devoted to Prolog and symbolic languages. In

particular, the use of the terms *top–down parsing, bottom–up parsing, inflectional morphology, derivational morphology*, and *word–formation morphology* might strike seasoned linguists and hardened Prolog programmers as unusual. I try to indicate those places where I differ from others and show why there are differences.

One major concern has been to provide a perspective internal to which one can conceptualize the aims and goals of linguistics in relation to the aims and goals of other sciences. To accomplish this, each of Chapters 6–8 has been written to define a perspective as well as illustrate the theories and methods of linguistic research.

Chapter 6 covers the type of material that was discussed in **taxonomic linguistics.** The goal of taxonomic linguistics, which basically refers to work prior to the ideas of Chomsky, is to obtain a redundancy–free catalog of the elements that function in a language and to indicate the restrictions on distribution of each element. All methods are designed to isolate and define the basic elements that function in a language.

Chapter 7 introduces the idea of **level**, which is basic to all varieties of linguistic analysis. The definitions of various types of morphology derive from the computational model presented here, which assumes that some morphemes (e.g., *be+en* of the passive, *wh*–elements of questions and relatives, etc.) are instructions to the parser about how to build the sentence. These morphemes are part of the logical relations of Prolog that define the combinations of elements at the level of syntax. Elements like *be+en*, *who*, *what*, *why*, and *how*, are instructions to the parser about how to access the lexicon for information about structure. These elements are only minimally represented in the lexicon (dictionary).

My view of the lexicon draws heavily on the work of Gross (1989a, 1989b, 1991, 1992, 1993), most of which describes complexities in French. I have followed his finite state model of the lexicon as closely as this presentation allowed. The lexicon for complement constructions (*expect, persuade, promise, want, seem...*) is essentially that presented in Leacock (1990, 1991). I also relied on insights provided in M. Baker (1988) and Stabler (1993).

In addition, Chapter 7 defines terms to facilitate exposition of basic Prolog tools (*append*, *recursion*, and *parallel processing*) to provide uninitiated readers with sufficient technical terms so that they can access traditional grammars (Jespersen, Curme, etc.) for more examples and to lead into my view of parsing. The parser considered here requires us to think of the elements (words, morphemes, formatives, stress levels, intonation patterns, etc.) in a sentence as having two functions: An element can be an instruction to the parser about how to access the lexicon. An element can be an item that does not contain specific instructions to the parser about the organization of the lexicon. Some elements are both. My definitions are consistent with my theory of lexical structure and model of parsing, but readers wedded to another framework may charge me with fitting morphology unhappily into a Procrustean bed.

O'Grady, Dobrovolsky, and Aronoff (1993) and Fromkin and Rodman (1993)

provide excellent introductions to morphology. Aronoff (1976) gives a detailed analysis of morphology. Sproat (1992) presents a computational analysis of morphology. A unification, or Prolog, approach to morphology is presented in Gazdar (1985), Koskenniemi (1983), and Tzoukerman and Liberman (1990).

Chapter 8 tackles what Chomsky has called Humboldt's problem: How can a human being make infinite use of finite means? I focus on **recursion –** a basic computational process that enables a Prolog program to call itself to define sentences. A grammar, represented as a **lexically driven parser**, assigns structures to sentences by projecting information about words (morphemes and formatives) given in the lexicon (dictionary).

Excellent textbooks covering the basic terminology of Chomsky's model include: C. Baker (1978), Freidin (1992), Lightfoot (1984, 1991), Radford (1985), and van Riemsdijk and Williams (1986). At a more advanced level, Lasnik and Uriagereka (1988) is excellent. These books do not cover computational models. Books that cover unification and Prolog in some detail, but that skimp on linguistic theory, include Pereira and Shieber (1987), Pereira and Warren (1980), and Shieber (1986, 1992). Those interested a detailed analysis of complex noun phrases should refer to Bains (1994), who presents a grammar of Hindi-Urdu. Meyers (1994) presents an analysis of various grammatical frameworks, including government and binding, in a unification framework. I rely heavily on his work.

Our study mainly presents Prolog grammars to encode Chomsky's ideas about the internal structure of simple sentences, that is, sentences containing only one verb, although I do present some Prolog grammars to describe complex structures. Dougherty (in preparation) presents Prolog grammars to describe complex coordinate sentences, for example, *Tess danced and Tracy sang, Neither when he was wise nor when he was unwise was Socrates ever boring*, and subordinate/complement constructions, for example, *Tess seems to be happy, Tracy is easy to please, Sean promised to be home on time, Tracy was expected to win*. To analyze these sentence structures one must understand how Prolog employs pointers to structure the memory, which is beyond this present study.

Semantics **–** which includes sense–disambiguation, ambiguity, synonymy, and so on **–** is discussed by Leacock et al. (1993a, 1993b), Miller (1991), and Miller et al. (1993, 1994) in the framework developed on the Word–Net Project (see Miller). I accept their concepts and definitions.

The text has been written to accord with three basic rules of style. Winston Churchill's dictum, "A preposition is something you should never end a sentence with," has been strictly observed. Bertrand Russell's insight: "And should only be used to begin a sentence," caused early drafts to be revised. The hardest rule has been that set by Charles S. Peirce: "Write for people who are intelligent, but who know nothing about your subject matter."

I want to thank all those students and colleagues who helped me with this

project. I feel particularly indebted to the following for their ideas, advice, and criticisms: Angel Arzan, Gurprit S. Bains, Scott Browne, Robert Corre, Lyle Jenkins, Peter Jeong, David Johnson, Thomas Kramer, Ira Langen, Gerald McCullum, Susanna Michalska, Dennis Perzanowsky, Amy Pierce, Paul Postal, Marc Schwarz, Steve Seegmiller, Linda Susman, George Thompson, Amy Weinberg, and Arthur Williamson. I thank Adam Meyers for his useful insights. Many of the ideas underlying my approach were developed while I was a Fullbright Scholar at the University of Salzburg (1976–1977) working with Gaberel Drachman, see Dougherty (1979). I thank Maurice Gross for introducing me to the use of finite state models in the lexical representation of natural language and for explaining to me many of the ideas of Zellig Harris. I thank David Halitsky for discussing all aspects of the book at length with me. I am especially indebted to Claudia Leacock for ideas, advice, and encouragement at every stage of the book's development.

I am grateful to Praxis International Inc. for funding research into the cross–point switch model of human language. The research into the cultural history of computers, robots, and artificial intelligence was funded by the New York University Humanities Council and by Exxon. I thank the New York University Academic Computing Facility for supplying the advice, information, and resources that made this project possible. I am especially indebted to David Ackerman, Ed Friedman, and Estelle Hochberg.