Book Reviews

T. Winograd, *Language as a Cognitive Process, Volume I: Syntax* (Addison-Wesley, Reading, MA, 1983); 640 pages, $34.95.

Reviewed by: Mihai Nadin
Advanced Computing Center for Art and Design, Cranston Center, 1501 Neil Avenue, Ohio State University, Columbus, OH 43201, U.S.A.

"Does this butcher knife
handle frozen foods, too?"

There was a time when children in grade school were required to parse, orally and on paper. Today, we study how to program computers to parse sentences in natural language and write books to introduce beginning graduate students to the complexities of computational linguistics. This change cannot be ignored when reviewing Terry Winograd's solid tome. He wanted it to be several things: textbook (including "detailed technical material and exercises designed to help the student master a body of concepts and techniques"); practical guide ("which should be useful in both the design and testing of systems" that deal with natural language); and reference source ("with many pointers to the literature of both linguistics and computer science"). Common sense requires one to consider a book in respect to the aims of its author. I doubt, however, that saying (and proving) that this is an excellent textbook and/or a successful practical guide and/or a comprehensive reference source is of any significance. Winograd is far from being a novice in the field. He is rightly credited for many innovative ideas (since his graduate studies at MIT in the AI laboratory) and deserves a challenging discussion of his work.

Mentioning that parsing used to be a classroom exercise, I intended to put the subject of the book in a cultural perspective. And I do so because the book lacks such a perspective. The two sections, "What every language user knows" and "The evolution of linguistic science" are but minimal reference alibis. The cultural perspective is necessary in order to understand *why* we ask questions about the syntactic level of language, and *how* such questions have changed through the years. The change of paradigms, which Winograd tends to embrace (although aware of Kuhn's rather sketchy model of *The Structure of Scientific Revolutions*), explains changes in the conceptual framework that scientists

Artificial Intelligence **27** (1985) 353–361
0004-3702/85/$3.30

apply in a given social structure but not in the larger framework of culture. It just happens that language is not a physical object, not a biological entity, and not so many other things that were/are traditionally associated with science. The peculiar nature of language—the most complex sign system we are aware of to date—imposes epistemological precautions that are not built into Kuhn's notion of scientific paradigm. Winograd's awareness of the complexities of the study of language is obvious, but he seems to ignore them in parts of the book. Distinguishing between the generative and the computational paradigms, Winograd successfully overcomes the oversimplification (calling it by its name) implicit in the model used. He points out similarities (whether they are principles or not remains to be seen) and differences. The decision to use the term "cognitive processing" for the common areas that they cover is indeed satisfactory, while the interchangeable use of the phrases "computational processing" and "cognitive processing" follows a stereotype.

Basically, I do not think that any valid approach to any system of signs (language, or systems of signs involved in what the author calls "non-linguistic knowledge") can start at the syntactic level, or can afford to adopt the autonomy of syntax hypothesis. What identifies language is the pragmatics, i.e., the functions it fulfills: communicative, social, aesthetic, cognitive, etc. These functions are not independent. But once a pragmatic perspective is identified, in respect to the particular function and from a synchronic (structuralist) perspective, it is possible to define the underlying syntax. While ascertaining that "Language is a process of communication between intelligent active processors, in which both the producer and the comprehender perform complex cognitive operations" (p. 13), the author uses a concept of communication so broad that it is no longer effective. His basic model preserves the dualistic, one-directional information theory representation, accepting the domination of sequences over configurations (time over space). The result is a hybrid between behaviorism and mentalism; consequently, a general equivocal attitude infiltrates the descriptions, the algorithms, or the effective procedures he suggests.

This book is important today because it shows that every computational approach is marked by the linguistic premise it adopts or by the lack of resolution in adopting such a premise. The material in the volume was "developed concurrently in linguistics and for engineering of computer systems." This explains the structure adopted. I think that the most convincing part of the book is "A Language for Describing Objects and Procedures" (Appendix A), in which the DL (Descriptive Language) is introduced. Winograd convincingly uses his concept of procedure. DL is based on the syntax of natural language and on some shared notation conventions. The attempt to achieve uniformity corresponds to the conceptual structure (not implementation). The device that would operate on programs generated from a DL perspective deals with *descriptions* (of linguistic objects). The focus is "on how

different data structures can best reflect the relevant structure of the object being described" (p. 419). Accordingly, *classes* of objects, logical *predicates*, and procedures (that operate on records describing objects of each class) represent the notions around which the entire notation was developed. DL is introduced to students with little background in computer science with the aim of allowing for the writing of programs and for their use on a computer. Including the definitions of numbers, characters, sequences, and sets in the "built-in class" definitions, DL is always procedure-oriented. The *Description* (definition, in plain English), *Roles* (other objects that can play a specific role associated to the class), *Background* (context), *Categories* (various partitions that pertain to the objects in the class), *Predicates* (for relationships involving objects in the class) participate in the stratified specification of a class in DL. Definitions of classes can be nested, Winograd exemplifies, as additional features, *Multiple roles* ("It is possible for an object to play more than one role with respect to another object," p. 423), *Computable roles and categories* (interdependency of attributes), *Fixed roles and categories* (marking as fixed), *Kinds*, (subclasses). This formalism provides for records that are filled according to a class definition describing objects of a class. The procedural dimension of DL, i.e., the set of forms and conventions for specifying the dynamics, involves *Procedure definition*, *Basic method* (sequence of steps to be taken), *Actions associated with primitives* (an *Iteration*, for instance, has three basic elements: *body*, *stepper*, and *stop test*), *Conditionals and case selection* (the body of the iteration), *Logical expressions* (as element of a conditional), *Record creation*, *Procedure variables* (inputs, results, working structures, background), *Declarations* (specifying variables), *Initial values*, *Embedded definitions* ("A procedure definition can contain further definitions to be used within it," p. 432). It is beyond the scope of this review to go into the details of this language, which is quite generously exemplified in the book. However, I cannot understand why Winograd did not place this appendix in the introductory part of his text since the DL formalism is used beginning with the first recognition procedure discussed ("Search for a match in a set of patterns," p. 37). Throughout the book, DL is the main tool used, and it proves to be a very clear and efficient descriptive methodology. How would a computer system using DL programs be designed? What kind of "smart compiler" will it require? How would debugging take place? These are questions that will be asked after Winograd's language receives the attention it deserves.

DL is pragmatically determined. Its semantics as well as its syntax result from the pragmatics assumed, Once established, DL proves a well-suited instrument for analyzing those syntactic aspects of language that Winograd is interested in. It also helps him describe some parsing procedures. A good example—from among the many very useful examples the book offers—is the active chart parser. The recognition that the overall approach to language and grammar is basic in the design of parsing procedure is a good premise of the section. General issues of

parallel versus sequential treatment of alternatives and top-down versus bottom-up analysis are precisely presented (unfortunately not the case with the section called "The problem of ambiguity"). The active chart parser is based on the definition of an algorithm that combines the advantages of both strategies while accomplishing "the two principles of efficient parsing: only do what is relevant; and don't do anything more than once" (p. 116). The record in such parsing is organized as a chart with a set of active edges (a coordinate in search of a constituent) and a set of complete edges (constituents found). The reader is informed that such an algorithm is very fast ("the amount of time taken to parse a sentence depends on the cube of its length if the grammar is ambiguous, and the square of the length otherwise," p. 126).

The trajectory from basic formalism for context-free grammars to the formalism developed along the line of transformational grammar is naturally extended to Augmented Transition Networks (ATN). Drawing from previous, known works, Winograd submits an augmented transition network formalism and applies it to parsing, dwelling on some syntactic aspects of English. Recursive Transition Newtorks (RTN), consisting of sets of labeled networks, can be used to parse sentences "with small modifications of the schemas and algorithms given for context-free grammars" (p. 198). Another chapter ("Feature and Function Grammars") brings a useful contrast to the cognitive paradigm, introducing formalisms based on *systemic grammar.* The opening to social aspects and to questions concerning social functions of language is definitely one of those decisions that originate in the pragmatics of language (in this case, the social function is chosen) and then pursued through the semantic level (indirectly) and syntactic level (in detail). The introduction to the subject that Winograd wrote in 1972 (*Understanding Natural Language*) and the program for language comprehension (SHRDLU) based on principles of systemic grammar are well known and make a discussion of the chapter superfluous. I expected that Halliday's book *Language as a Social Semiotic* (1978), and the semiotic paradigm in general, would have inspired some changes, if not new directions in research. Would it be possible to use DL for feature and function grammar? I missed any reference to this topic. After the concepts related to syntax and parsing are introduced, systems developed between 1950 and the days in which the book was being finished are presented. Some are discussed in depth; others are merely mentioned. It is perhaps the most comprehensive survey available and once again proof that syntax issues can hardly be approached independently of semantics and pragmatics. Winograd knew this before writing the book. Commentators of his work are unanimous in giving him credit for this. I assume that the second volume will re-establish the fine balance we must keep while studying and practicing all kinds of computer applications for natural language. But once this point is reached, a final basic question arises: Aren's we too hasty in saying that "The computer shares with the human mind the ability to manipulate symbols and

carry out complex processes that include making decisions on the basis of stored knowledge" (p. 13)? Isn't it rather a way of speaking? Digital or analogic devices do not manipulate symbols, only *representations* of symbols. Writing this, I contradict statements that I myself made several times, feeling so sure that the computer is more than a "number-crunching" machine. But from time to time, there is a need to re-evaluate some of our statements, to go beneath the surface, and to use more precise concepts, a better formalism. Ignoring basic work in semiotics (or perhaps unaware of it), computer scientists and AI scholars operate with a very loose concept of symbol. The looseness of the concept used makes the statement about the ability to manipulate symbols almost irrelevant. Even the concept of natural language is worth a better definition. Actually several kinds of language, and several forms of intelligence, should be considered. Boole's *Investigation of the Laws of Thought* (1854) is definitely more precise than we sometimes are in identifying constraints inherent in his logic. The need for reassessment concerns not only concepts, but also some directions. Explaining regularities within linguistic structures is of major importance. So is the attempt to integrate linguistic knowledge in machines able to perform well defined tasks. But fundamental research (Winograd's book, as a textbook, is perhaps not the best place to look for it) has to go beyond the set of confirmations acquired and to *point to critical issues.* A student's mastery of the confirmed procedures of computational processing may result in good (and necessary) operators of (rather primitive) available systems. A student's awareness of critical issues might trigger work that could change our rudimentary heuristics (touching "only the tiniest bit of the relevant knowledge", cf. Winograd, *A Procedural Model of Language Understanding,* 1973). Winograd's excellent book is one more proof that such changes are inevitable.

G. Lakoff and M. Johnson, *Metaphors We Live By* (Chicago University Press, Chicago, IL, 1980); 239 pages, $8.95.

Reviewed by: Peter Norvig
Department of Computer Science, 573 Evans Hall, University of California at Berkeley, Berkeley, CA 94720, U.S.A.

Wayne Booth [1] has written that, judging from the recent jump in interest in metaphor, if we extrapolate to the year 2039, there will be more students of metaphor than people. Linguists, philosophers, and psychologists have been quick to jump on the metaphorical bandwagon, but so far AI researchers have not. Lakoff and Johnson's *Metaphors We Live By* (henceforth *MWLB*) is an