Constraint-Based Grammar Formalisms: Parsing and Type Inference for Natural and Computer Languages

Stuart M. Shieber
(Harvard University)

Cambridge, MA: The MIT Press, 1992,
xi + 183 pp.
Hardbound, ISBN 0-262-19324-8, $24.95

Reviewed by
Veronica Dahl
Simon Fraser University

1. Introduction

This excellent book addresses the difficult and interesting task of providing a single theoretical and computational framework for the various grammar formalisms that have been characterized as constraint-based, information-based, or unification-based. The variety of such formalisms that resulted from their having independently sprung from different fields—linguistics, computational linguistics, and artificial intelligence—makes it difficult to establish useful comparisons between them. The author identifies some of their common threads, such as the use of declarative constructs, the modularization of information, and the crafted manipulation of partial information that eventually fits together with other partial information in order to become complete.

It is the book's objective to "build some of the foundational understanding of this class of formalisms—from both a mathematical and a computational perspective." In so doing, Shieber uncovers many implicit conceptions of information and constraints that are involved in these formalisms—hence the preferred denomination "constraint-based."

Rather than defining any one particular conception of information, Shieber abstracts from those conceptions implicit in the various previous approaches a notion of information defined in terms of its desired properties, e.g., modularity (information must be partitioned into different, declarative modules), partiality (information does not need to be complete at all times, but can arise from the combination of partial information available from a variety of sources), and equationality (informational constraints interact to place strong limits on the distribution of a phrase, rather than giving information about that phrase directly).[1] For instance, in the PATR formalism, modules are graph structures (rules), only the partial information available at each inductive step is noted, and informational constraints are given as equations.

At a time in which constraint-based reasoning is ubiquitous in many branches of science, including in the field of computational linguistics, we must hasten to add that the notion of constraint examined in Shieber's work is quite different from the notion of constraint satisfaction as originally described by Waltz (1975). Rather, we are looking at constraints such as equational ones—whether they are described as explicit

1 The use of the term *equationality* is a bit of a synecdoche. The need for types of constraints other than equations is explicitly taken into account in Shieber's work, but equational constraints are given such prominence that their denomination stands, somewhat abusively, for the whole property's denomination.

equations or as variable sharing—or constraints requiring the existence of values, set membership constraints, etc. Such constraints are present in formalisms such as Augmented Transition Networks (Woods 1970), Lexical Functional Grammars (Bresnan 1982), Functional Unification Grammar (Kay 1984), logic grammars as evolved from Metamorphosis Grammars (Colmerauer 1978), grammars in the Generalized Phrase Structure Grammar framework, as evolved from Gazdar (1982), and the PATR formalism (Rosenschein and Shieber 1982).

Having replaced the notion of information by an abstraction in terms of its properties, a class of logics of information with rigorous semantics is then proposed to codify these properties. Given these logical systems, a grammar formalism can be described on the basis of logical constraints over information associated with phrases. Further properties of the models of the logic are postulated so that a general algorithm can be defined for parsing phrases with respect to these grammars. Proof of the correctness of this algorithm is given independently of both the logics upon which the formalism is based and the control regime used. Particular instances of the algorithm include variants of well-known parsing algorithms such as Earley's and LR parsing.

2. Summary and Discussion of the Book

The intended audience is computational linguists, theoretical linguists, and computer scientists, but the book is also of interest to logicians and semanticists.

Chapter 1 provides a very brief, intuitive introduction of the work, highlighting the questions to be addressed in the book, the methodology chosen, and the results and interesting byproducts of the author's attempt to characterize the abstract notion of a constraint-based grammar formalism.

Chapter 2 includes, in a sense, what could have been the real introduction. While its stated objective is to develop a system of constraint logics for linguistic information from a purely abstract starting point, the chapter plunges nonetheless nearly immediately into introducing the concrete PATR formalism,[2] which serves throughout the book as an exemplifying anchor. Then it proceeds to introduce a notion that will be developed later; namely, that the book's approach to constraint-based formalisms has attractive implications for the description of computer languages. It also presents a history of constraint-based formalisms and of related formalisms and languages from computer science. Only then does the chapter get into its subject matter, by proposing a class of logics that can accommodate the abstract properties of information and constraints that are singled out as necessary for all the formalisms considered. Great attention is given to the development of such systems from a desiderata of properties, and to the rigorous statement and proofs of all properties that the resulting logic systems possess. The notation is in some places ambiguous (angle brackets are used both for paths and for substitutions), and some of the concepts used are not clearly defined (e.g., p. 27 should include a clear definition of what it means for a formula to be defined). But taken as a whole, the formulation is a very appealing one.

Chapter 3 defines constraint-based grammar formalisms in terms of particular logical systems of the type developed in Chapter 2. Again, PATR is taken as the exemplifying focus. An abstract parsing algorithm is presented that can be viewed as

2 It is unfortunate for those readers without a background in PATR that some mistakes have crept unnoticed into the very presentation of the formalism, but the self-confident reader can detect them by common sense even without a background. For example, the arc labels *cat* and *agr* are interchanged in two examples, which results in a VP being labeled *agr* rather than *cat*; rule R4 on p. 11 should express agreement between the zeroth and the first argument rather than the first and the second as is stated.

an abstraction, done in terms of the logic developed, of table-based algorithms such as Knuth's LR (Knuth 1965), Earley's algorithm (Earley 1970), or chart-parsing (Kay 1980). This algorithm is independent both from a particular formalism and from the control regime, and beautifully concise. It is based on structures called *items*, which encode information about contextually allowed partial phrases, and contains four inference rules: a rule that builds up the initial item; a scanning rule, which parses the next word in the string; a prediction rule, which tries rules that may parse the next unparsed substring; and a completion rule, which combines two compatible partial parses into a single item. The algorithm is also reminiscent of some logic grammar formalisms, such as puzzle grammars (Sabatier 1984), or of extensions of context-free formalisms such as TAG (Joshi 1985), in that it can be visualized as "gluing together" partial parse trees to obtain a final one. The gluing is in this case formalized in terms of three operations defined on models: union, extraction, and embedding. Important formal properties of the algorithm are proved—in particular, its partial correctness and completeness. Instances with respect to which it is possible to eliminate the redundant generation of items, to mitigate nontermination problems, or to specify a given control regime are also discussed.

Chapter 4 examines appropriate classes of models for the constraint logics. The logical systems based on these models are required to possess independent properties: denotational soundness, logical soundness, logical completeness, minimal-model existence, and categoricity. In addition, computational efficacy requires the properties of finiteness of minimal models of atomic formulas, finiteness preservation for model operations, and computability of operations. Starting from finite-tree models, the author formally argues the inappropriateness, with respect to these desired properties, of various successively proposed models, and finally arrives at graph models, which can provide a complete model structure for the logic (unlike finite tree models), can distinguish intensional from extensional identity, and allow for the computational efficacy of the parsing algorithm. This chapter is perhaps the one that relies the most on formal background on the reader's part. It assumes a logical background that is not so crucial in the other chapters, which always include intuitive notions of the formalized concepts as well. This is probably inevitable in the context of the book, which covers so wide a spectrum. But the chapter would benefit from extra attention to clarity of presentation (e.g., on p. 89, the extension of the function application notation to apply to paths as well as labels is somewhat unclear).

Chapter 5 discusses the fascinating topic of transferring the constraint-based framework developed into an entirely different field, that of type inference in computer languages. The constraints are extended to encompass inequations as well as equations. Such constraints can be viewed as filling the role of typing rules for a programming language (in the sense of type inferencing according to explicit rules in a deductive calculus of types). In this view, the notions of category from linguistic theory and type from computer science can be identified. The connections between computer languages and the inequality logic are discussed in the light of the class of models ensuing from the incorporation of inequations, always respecting the foundations developed in the previous chapter. A constraint grammar to perform type inference for a simple applicative programming language is evolved and its shortcomings are also described. While speculative, this chapter raises interesting possibilities of mutual feedback between solutions to natural language and programming language problems. For instance, the extensions of the kinds of constraints in Shieber's framework that are motivated by problems of binding and polymorphism in typed programming languages turn out to be applicable to some natural language description problems, such as nonmonotonic constraints and coordination.

Chapter 6 is a brief summary of the novel results presented in the book, and also stresses its significance as an initial effort toward a nontraditional approach to designing grammar formalisms: instead of first designing the data structures to be used for linguistic information representation and then constructing a language around them, the appropriate data structures or models are developed on the basis of prior desiderata on the more abstract level of which kinds of information (i.e., information with which properties) are wanted.

3. Summary Critique

Shieber's book represents a fascinating approach to uncovering the theoretical foundations of a very important class of grammar formalisms. The desire for a uniform theoretical account for all of them, while quite ambitious, leads to very interesting results, which are applicable also outside the realm of constraint-based grammars.

The methodology chosen—that of developing models based on prior desiderata, as opposed to the more contingent ways in which the existing formalisms evolved—seems in some cases to give way, despite the best intentions, to perhaps excessive influences from the idiosyncrasies of some of these very formalisms. For instance, the anchoring on PATR, while providing clear exemplification of the concepts presented, also seems to unduly determine to some extent which concepts are stressed.

This is often the case in efforts to give theoretical foundations to existing formalisms, and does not detract from the beauty of Shieber's work. But it would be interesting to examine to what extent Shieber's proposal can be adapted to provide foundations to formalisms that are also in the constraint-based class as defined in the book, but belong to different schools of thought than those having inspired it. For instance, linguistically principled approaches in the Chomskyan tradition have also given rise to a variety of constraint-based formalisms, which might not fit naturally foundations that stress equationality (e.g., Johnson 1991a; Stabler 1992; Dahl, Popowich, and Rochemont, in press), or that promote, as Shieber's do, a conflating of syntax proper and typing constraints. In some approaches, phrase structure constructions are explicitly separated from the application of type-inferencing rules, where the notion of type refers to types of linguistic principles, implemented as well-formedness conditions (e.g., Fong 1991). Taking into account logic-based approaches to the quest for a common language in which constraint-based grammar formalisms can be expressed (e.g., Carpenter 1992; Johnson 1991b) might also be interesting from the point of view of placing the present work in wider contexts.

Despite these and other minor reservations I have pointed out, I am very impressed by how much the book accomplishes with respect to the tremendously ambitious task it sets out to perform. Shieber's analyses and formulations are elegant and thorough, they illuminate many aspects of an important problem, and the presentation makes most pleasant and stimulating reading. I enthusiastically recommend this book to everyone interested in computational or theoretical linguists.

References

Bresnan, Joan (editor) (1982). *The Mental Representation of Grammatical Relations*. The MIT Press.

Carpenter, Bob (1992). *The Logic of Typed Feature Structures: With Applications to Unification Grammars, Logic Programs, and Constraint Resolution*. Cambridge University Press.

Colmerauer, Alain (1978). "Metamorphosis grammars." In *Natural Language Communication with Computers*, edited by Leonard Bolc. Springer-Verlag.

Dahl, Veronica; Popowich, Fred; and

Rochemont, M. (in press). "A principled characterization of dislocated phrases: Capturing barriers with static discontinuity grammars." *Linguistics and Philosophy.*

Earley, J. (1970). "An efficient context-free parsing algorithm." *Communications of the ACM,* 13(2), 94–102.

Fong, Sandiway (1991). "Principle-based parsing and type inference." *Proceedings of the Third International Workshop on Natural Language and Logic Programming,* edited by C. G. Brown and G. Koch.

Gazdar, Gerald (1982). "Phrase structure grammar." In *The Nature of Syntactic Representation,* edited by Pauline Jacobson and Geoffrey K. Pullum, 131–184. D. Reidel.

Johnson, M. (1991a). "Deductive parsing: The use of knowledge of language." In *Principle-Based Parsing: Computation and Psycholinguistics,* edited by Robert Berwick, Steven Abney, and Carole Tenny. Kluwer Academic Publishers.

Johnson, M. (1991b). "Logic and feature structures." In *Proceedings, 12th International Joint Conference on Artificial Intelligence.* Sydney, Australia, 992–996.

Joshi, Aravind K. (1985). "Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?" In *Natural Language Parsing,* edited by David Dowty, Lauri Karttunen, and Arnold M. Zwicky, 206–250. Cambridge University Press.

Kay, Martin (1980). "Algorithm schemata and data structures in syntactic processing." Technical Report, Xerox Palo Alto Research Center, Palo Alto, CA.

Kay, Martin (1984). "Unification in grammar." In *Natural Language Understanding and Logic Programming,* edited by Veronica Dahl and Patrick Saint-Dizier, 233–240. North-Holland.

Knuth, Donald E. (1965). "On the translation of languages from left to right." *Information and Control,* 8(6), 607–639.

Rosenschein, Stanley J., and Shieber, Stuart M. (1982). "Translating English into logical form. " In *Proceedings, 20th Annual Meeting of the Association for Computational Linguistics.* Toronto, Canada, 1–8.

Sabatier, P. (1984). "Puzzle grammars." In *Natural Language Understanding and Logic Programming,* edited by Veronica Dahl and Patrick Saint-Dizier, 139–152. North-Holland.

Stabler, Edward P. Jr. (1992). *The Logical Approach to Syntax: Foundations, Specifications and Implementations of Theories of Government and Binding.* The MIT Press.

Waltz, David (1975). "Understanding line drawings of scenes with shadows." In *The Psychology of Computer Vision,* edited by Patrick Winston. McGraw-Hill.

Woods, William A. (1970). "Transition network grammars for natural language analysis." *Communications of the ACM,* 13(10).

Veronica Dahl co-authored the book *Logic Grammars* with Harvey Abramson (Springer-Verlag, 1989), and has widely promoted with her writings the uses of logic in general and logic programming in particular for language processing and for knowledge-based systems. She is the area editor for Formal and Natural Languages for the *Journal of Logic Programming.* Dahl's address is: Computing Sciences Department, Simon Fraser University, Burnaby, B.C., V5A 1S6, Canada; e-mail: veronica@cs.sfu.ca.