Book Reviews

Representation and Inference for Natural Language: A First Course in Computational Semantics

Patrick Blackburn and Johan Bos
(INRIA, France, and University of Edinburgh, Scotland)

Stanford: CSLI Publications (CSLI Studies in Computational Linguistics, edited by Ann Copestake), distributed by the University of Chicago Press, 2005, xxv+350 pp; paperbound, ISBN 1-58576-496-7, $30.00, £21.00

Reviewed by
Francis Jeffry Pelletier
Simon Fraser University

Computational semantics is the study of how to represent meaning in a way that computers can use. For the authors of this textbook, this study includes the representation of the meaning of natural language in logic formalisms, the recognition of certain relations that hold within this formalization (such as synonymy, consistency, and implication), and the computational implementation of all this. I think that, while there probably are not many courses devoted to computational semantics, this book could profitably be incorporated into more traditional computational linguistics courses, especially when two courses are offered serially. The material here could be spread out and integrated into parts of a more standard pair of these courses, and it would result in a substantial widening of the knowledge that students come away with from these courses.

The introduction of this book traces the history of computational semantics, with a goal of justifying the enterprise in the face of the modern emphasis on statistical natural language processing. Besides this introduction, the book contains six substantial chapters and four short appendices. There is also a very extensive suite of material online at a Web site maintained by the authors.

Chapter 1 is an introduction to first-order logic, but with a twist that underscores the particular outlook taken in this book. There is a *very* short introduction to the notion of a model of a set of quantifier-free sentences, followed by an equally short discussion of the interpretation of quantifiers in a model. With respect to quantifier-free sentences, students are referred to the three-page Appendix B on propositional logic, where truth-tables are covered. The innovations start with the introduction of three "inference tasks": querying, consistency checking, and informativity checking. These correspond to the logical concepts of satisfiability (of a given formula in a given model), consistency (whether there is a model that satisfies a given formula), and validity (whether a formula is true in all models, or, equivalently, whether a given argument is valid). Although the notions of querying, etc., are thus merely renamings of standard logical notions, it seems to me that the renaming is particularly apposite in the setting of a textbook for non-logicians, especially since much is to be done computationally with these notions in the enterprise of computational semantics. Much of the remainder of chapter 1 consists of a gentle introduction to Prolog and using it to check formulas for well-formedness and to build model checkers with the goal of implementing the querying task.

Chapter 1 closes with a short discussion of the shortcomings of first-order logic as a representational language, which serves to motivate chapter 2, "Lambda calculus." This chapter starts with the question of whether we can automate associating semantic representations with sentences of natural language. A "first pass" is made in terms of definite-clause grammars, and some "experiments" are carried out to show students the necessity of having some representational scheme similar to the lambda calculus. This seems to me a very nice motivation for lambda calculus, and the remainder of this chapter is developed with an eye to simultaneously doing "language engineering," "lexical development," and "semantic rule manipulation." Some interesting exercises guide the student along this path.

Chapter 3 introduces underspecified representations, with particular attention to scope ambiguity. Four methods are considered in turn: Montague's original method, Cooper storage, Keller storage (i.e., nested Cooper storage), and hole semantics. Montague's method of generating different scopes by generating their analysis trees differently is described in some detail and found "inelegant." This leads to the other three methods, for which implementations in Prolog are built up, starting with Cooper storage. The authors remark that Cooper storage is extremely easy to implement, but runs into trouble with "nested NPs," such as *Every owner of a hash bar*, where the Cooper method inappropriately generates unbound variables for some retrievals of the NPs in the store. This motivates the Keller method, which is essentially just Cooper storage with a method for ensuring that the embedded NPs retain appropriate values for their variables that are contributed from the embedding NPs. The authors remark that Keller storage is "a very simple modification of our earlier code for Cooper storage."

However, these storage mechanisms are found wanting. Sometimes we might want to say that one NP must definitely outscope another, while at the same time saying that they have no scoping relation to a third NP. Further, there are other constructs (e.g., negation and intensional verbs such as *knows that*—although the intensional constructions are not considered in this book) that introduce scope ambiguities, and the NP storage method seems unable to generalize to these cases. The authors thus motivate the more general notion of *underspecified semantic representation* (USR). The particular version of USR they adopt is (not surprisingly) Johan Bos's *hole semantics*. This is the longest (and most intricate) part of the chapter, and no doubt the place where students will most easily stumble. The transition to hole semantics is aided by Prolog macros for building USR trees with the appropriate labeling, and this is followed by a number of plugging predicates, also written in Prolog. By the end of the chapter, and by using the on-line materials, students can come to manipulate moderately complicated USRs.

Chapters 4 and 5 bring in a consideration not normally contemplated in computational linguistic courses: inference. Recall the authors' desire to have (a) querying, (b) consistency checking, and (c) informativity checking as a part of their computational semantics. (Indeed, they seem to claim that this is a primary source of data against which one can evaluate the worth of (computational) semantic theories.) These are to be implemented as (a′) model checking, (b′) model building, and (c′) validity of arguments. They therefore wish to involve automated methods of performing these tasks. To these ends, they introduce automated theorem proving at length. (Automated model checking is an easier task that was discussed in chapter 1.) With respect to the propositional logic (chapter 4), signed tableau and resolution methods are introduced, first in theory and then with respect to some Prolog macros. Students are encouraged to experiment with some provers that the authors have on-line. The section concludes with a discussion of some meta-theoretic properties of propositional logic and remarks about issues of complexity. Chapter 5 expands both tableau and resolution to the first-order

case. This chapter contains a lot of information about technical issues in automated theorem proving. After developing all this machinery, the authors surprise the student by tossing away all the theorem provers that have been built, on the twin grounds that "they are too naïve" and "they don't handle equality." Instead, the authors move to off-the-shelf theorem provers, in particular to Otter and Bliksem, which are both resolution-based. The authors have written interfaces that allow students to enter formulas in the notation employed in the book and call up one or another of these provers (which are on the book's Web site). Also on the Web site are the model-building programs Mace and Paradox, and although there was no discussion in the text about how model building is automated, students can use these programs to build models for sets of formulas.

Chapter 6 is entitled "Putting It All Together," and it is here that the lambda calculus, the USRs, the theorem-proving programs, the model-checking programs, and the model-building programs are employed together to construct a system, "Clever Use of Reasoning Tools" (Curt). Curt is developed in seven stages, each one adding some aspect that was discussed earlier in the text to the preceding version. The final version, Knowledgeable Curt, has a small fund of lexical knowledge, world knowledge, and situational knowledge at its disposal. And like some of the earlier Curts, it can maintain a model of the Discourse-So-Far. On the basis of all this, Knowledgeable Curt can accept new information if it does not contradict what is already known or is not implied by what is already in its store of information. And for simple dialogues, it does a credible job. This leads into the last part of the book, which is a discussion of lexical knowledge, world knowledge, ontologies, and the like. Despite its simplicity, Curt is a "proof of concept" for the entire enterprise being undertaken in this textbook, and students who actually get this far and can work with Curt will be interested in extending it. And isn't that what we would want out of any course?

I have not yet mentioned the "notes" sections at the end of each chapter. These are always interesting and will give the instructor who uses this book a much broader understanding of the topics under discussion in the chapters. They are also quite nice for historical and bibliographic information.

Choosing material to put into a textbook is always a matter of juggling priorities, and so it will always lead others to complain that certain things were omitted that should not have been and that too much time was spent on some other things. This book, with its unusual emphasis on (a) semantics without having any theoretical discussions to motivate a firm syntactic base, (b) first-order logic without considering any other framework, (c) underspecified semantics over other alternatives, (d) hole semantics as the favored version of underspecified semantics, and (e) theorem-proving technology in the service of semantics, is bound to generate complaints. I would have liked to see some firm syntactic theory or other that would form the basis for generating semantic representations, with thoughts on how to arrive at such a grammar. (This book and its Web site present a small context-free grammar.) I would have liked to see at least a mention of alternatives to underspecified semantics, and possibly some general criticisms of underspecified forms (perhaps in the notes). And even though some of my own work is in automated theorem proving and its use in semantics, it is not so clear to me that the discussion of the inner workings of tableau and resolution methods is really appropriate here, especially since the systems that are developed are discarded in favor of off-the-shelf systems. Furthermore, the authors also employ automated model-building programs, and these are not explained at all, in contrast to the theorem-proving methods. In the model-building case, students are told from the beginning to just use the off-the-shelf software; why not do the same with the theorem provers? Much could

have been omitted from these chapters on theorem proving (and replaced with more straightforward semantic information) if theorem proving weren't covered so deeply.

Still, these are quibbles. We have here perhaps the only book in its class: an exploration of computational semantics aimed at the senior undergraduate student or beginning graduate student who has not taken any computational semantics. It nicely introduces a current influential direction in the research field and might even convince students that there is more to semantics than corpus-based studies of what words are *n*-grams of some given word. It might also have some positive effect on automated theorem proving, moving its current emphasis on mathematical issues to include language-oriented topics.

Francis Jeffry Pelletier is a Canada Research Chair in Cognitive Science and Professor of Philosophy and of Linguistics at Simon Fraser University. His research is in philosophy of language and logic, with emphases on both human performance and on computational implementation. He has also done work in automated theorem proving and ancient Greek philosophy. He has been editor, associate editor, and on the editorial boards of a wide range of journals (including *Computational Linguistics*), and is an editor of the Springer book series *Studies in Linguistics and Philosophy*. Pelletier's address is Department of Philosophy, Simon Fraser University, Burnaby, BC, Canada V5A 1S6; e-mail: jeffpell@sfu.ca.