Reversible Grammar in Natural Language Processing

Tomek Strzalkowski (editor)
(New York University)

Boston: Kluwer Academic Publishers
(The Kluwer International Series in
Electrical Engineering and Computer
Science; Natural Language Processing
and Machine Translation, edited by
Jaime Carbonell), 1994, xxi + 454 pp.
Hardbound, ISBN 0-7923-9416-9,
$130.00, £94.50, Dfl 265.00

Reviewed by
S. G. Pulman
SRI International, Cambridge, and University of Cambridge

This is a collection of 15 revised and expanded papers from the 1991 Berkeley ACL workshop of the same title. Collectively, the contributions cover all the main theoretical and practical motivations for reversibility of grammars: increased accuracy of description gained; economy of development and maintenance effort; consistency of language use in translation or interface applications; and possibility of self-monitoring for ambiguity in generation or paraphrase of parsing input. Some other connected issues are also discussed.

"A reversible constraint-based logic grammar," by Palmira Marrafa and Patrick Saint-Dizier, describes a formalism derived based on Aït-Kaci's psi-terms, and shows how the operation of type construction can be used both to generate and to parse types representing sentences and their meanings. Some illustration of the formalism is given by applying it to problems of Portuguese syntactic description.

Marc Dymetman, in "Inherently reversible grammars," distinguishes different ways in which "reversibility" can be understood. He focuses on a notion of "inherent finite reversibility" of grammars: a property possessed by a grammar if it is possible to derive sound, complete, and terminating parsing and generation algorithms for it. He discusses some grammars of artificial languages that do not satisfy this condition and tries to define a property of "moderation" that distinguishes the two classes of grammars. The intuitive content of this is something like compositionality.

Gunter Neumann and Gertjan van Noord ("Reversibility and self-monitoring in natural language generation") point out that a reversible grammar is necessary for self-monitoring for ambiguity in generation, or for paraphrasing of ambiguity when parsing. They describe an algorithm for carrying out both of these tasks, illustrated mostly with examples of PP-attachment ambiguity. I found this one of the most interesting papers in the volume. However, there is an important practical issue not discussed by the authors that arises when such techniques are used in practical applications, namely that in many cases multiple readings are not distinguishable by a nonexpert. For example, each of the following has at least two parses but most nonlinguists (and some linguists) are incapable of discerning any difference in meaning:

> I'll pick up the car at the airport.
> Can I make a stop in Boston?

Under such circumstances, techniques for the avoidance of ambiguity create a problem instead of solving one.

Rémi Zajac ("A uniform architecture for parsing, generation, and transfer") describes a typed-feature-logic implementation that allows both for analysis and generation as well as for bidirectional transfer. The system is extremely elegant and truly nondirectional. Reading between the lines (p. 108), one observes that this lack of directionality also means that it is equally inefficient in any direction, and Zajac remarks that the system should be seen as a grammar development tool rather than as something on which to base practical applications. In an ideal world, one would like to see the emergence of compilation and optimizing techniques akin to those described elsewhere in this volume for this class of formalisms.

In "Handling felicity conditions with a reversible architecture," Masato Ishizaki formalizes a type of planning operator to capture linguistic felicity conditions using the notation of HPSG. It was not clear to me that any immediate advantages accrue from this notational exercise, since—if I have understood correctly—the usual operational semantics assumed for HPSG-like formalisms is no longer assumed. Feature equations representing preconditions that might not be able to be satisfied when they are encountered are apparently intended to be treated like constraints, and "processing continues to hold them as assumptions. They are passed up or down along derivation" (p. 124). I would have welcomed some more clarification of the motivation for this way of handling felicity conditions.

Koiti Hasida ("Common heuristics for parsing, generation, and whatever . . . " [sic]) describes a constraint-solving method for first-order logic problems that is claimed to be able to simulate some well-known parsing or generation strategies. I regret to say that I found the details too unfamiliar and hard to follow in the time I could give it to be able to assess this ambitious claim properly.

Hans Ulrich Block ("Compiling trace and unification grammar") describes a unification-based formalism that contains special notation for "movement rules." The formalism is compiled in different ways for parsing and for generation. Block describes the various stages of compilation, which lead to very efficient run-time systems while still allowing the linguist to use a high-level notation. I found this a very clear paper describing a good piece of computational linguistic engineering.

Tomek Strzalkowski ("A general computational method for grammar inversion") describes what is presumably the final and definitive version of his procedure for inverting logic programs. The method is illustrated with admirable detail and clarity. However, in the context of most of the discussion in this book, the starting point for this algorithm is a peculiar one, namely a "parsing-oriented unification grammar," and furthermore, a grammar that is conceived of (like a DCG) as being executed as a program. But do people write grammars like this these days? It would be nice to see procedures similar to those described here allowing for efficient processing with grammar formalisms such as those described by Zajac, or Marrafa and Saint-Dizier. Grammars in such formalisms are not (intentionally) parsing- or generation-oriented. For development purposes, the grammarian needs to see what is happening in a form as similar as possible to the way the grammar was written. I would guess that trying to debug either the input or output grammars of Strzalkowski-ization has all the charm of reading core dumps: an optimal system would allow grammar development to take place using a high-level notation, with the kind of grammars described by Strzalkowski being the output of a compilation phase.

James Barnett, in "Bi-directional preferences," discusses the issue of applying preferences, and combinations of preferences, as a disambiguation heuristic during parsing and generation.

Lee Fedder ("Handling syntactic alternatives in a reversible grammar") adds some features indicating information structure properties to a unification grammar. This simple device is quite effective in ensuring that answers to questions are generated in a form that is appropriate to the way the question was asked.

David McDonald ("Reversible NLP by linking the grammar to the knowledge base") describes a system that compiles TAGs for analysis and generation, and also links this with a semantic model. The resulting system looks sufficiently different from TAGs that McDonald devotes some attention to the question of whether the compilation preserves equivalence.

Dominique Estival ("Reversible grammars and their application in machine translation") describes an implemented unification-based analysis, transfer, and generation system. Efficiency problems caused by asymmetries of information flow in different processing directions are alleviated by declarations that the grammar writer can make.

In "Reversible machine translation: What to do when the languages don't match up," James Barnett, Inderjeet Mani, and Elaine Rich discuss translation mismatches, beginning with a good survey of well-known translation problems. They then present an algorithm (unimplemented, at least on the evidence of this paper) for solving some of the mismatches that arise through problems of lexical choice. This procedure presupposes a reversible grammar.

The final two papers are linked. Robin Fawcett's paper ("A generationist approach to grammar reversibility in natural language processing") is a long discussion amplifying the point that, because systemic grammar cuts the syntax–semantics–pragmatics pie differently from other approaches, the notions of reversibility, parsing, and generation also differ. Tim O'Donoghue's companion paper ("Semantic interpretation in a systemic functional grammar") discusses the mechanisms of semantic interpretation in systemic grammar, and mentions some issues that arise when reversibility is taken seriously. I am afraid that neither paper did anything to dispel my (no doubt unfair) prejudice that systemic functional grammar has the strongest current claim to be the FORTRAN of linguistic formalisms.

Should you buy this book? For me, it falls into the category of those books that I would be happy to recommend the librarian to get, but less willing to spend my own money on. About 30% of the papers are perhaps perfectly worthy in themselves but not very directly related to the main theme. A more ruthless pruning might have lost the editor some friends, but would have resulted in a more tightly focused (and cheaper?) book.

Nevertheless, I learned something from almost every paper in this collection, and I'm sure I will go back to a couple of them more than once.

Stephen Pulman is a lecturer at the University of Cambridge Computer Laboratory and Director of SRI International's Cambridge Computer Science Research Centre. He has previously worked in computational morphology, syntax, and parsing. His current research interests are in semantics and dialogue, and, in particular, in the development of descriptions of the meanings of contextually dependent constructs that allow for reversible processing. Pulman's address is: SRI International, 23 Miller's Yard, Mill Lane, Cambridge CB2 1RQ, England. E-mail: sgp@cam.sri.com.