Book Reviews

Parsing Schemata for Practical Text Analysis

Carlos Gómez Rodríguez
(University of A Coruña)

London: Imperial College Press (Mathematics, computing, language, and life series, edited by Carlos Martin-Vide, volume 1), 2010, xiv+275 pp; hardbound, ISBN 978-1-84816-560-1, $89.00

Reviewed by
Anoop Sarkar
Simon Fraser University

Deductive systems are a widely used exposition technique in contemporary computational linguistics to explain novel parsing algorithms (e.g., Huang and Sagae 2010) and decoders in machine translation (e.g., Huang and Mi 2010). Deductive parsing provides a succinct formal syntax that abstracts away the implementation details yet directly reflects the time and space complexity of the underlying algorithm.

A deductive system can be viewed as a domain-specific declarative schema that specifies a parser at an abstract level, focusing on the semantics of the parser actions rather than implementation. The schema itself can be compiled into an executable parser allowing implementation optimizations to be shared across different parsing algorithms. Schemas thus provide quick prototyping of parsing algorithms. In the notation used by the compiler described in this book we would describe the familiar CYK parsing algorithm (Kasami 1965; Younger 1967) as the following declarative specification:

$$\begin{array}{l} @\textit{step Unary} \\ \dfrac{[a, i, i+1]}{[A, i, i+1]} \; A \rightarrow a \\ @\textit{step Binary} \\ \dfrac{\begin{array}{l}[B, i, j] \\ [C, j, k]\end{array}}{[A, i, k]} \; A \rightarrow B\ C \\ @\textit{goal}\ [S, 0, \textit{length}] \end{array}$$

Deductive systems can provide a framework to prove correctness (soundness and completeness) of a parsing algorithm. Well-formed transformations of deductive systems would permit the addition of new capabilities such as weighted rules in the grammar. Deductive systems could also provide a means to compare different parsing algorithms. This book provides several examples of how such properties can be useful in parsing theory and parsing implementation, in particular for converting a parser into an error-correcting parser and explicitly showing the relationship between several dependency parsing algorithms.

Sikkel's definition of parsing schemas (Sikkel 1997) extends deductive systems by formally defining the semantics of items and related concepts used in deductive systems. In particular, items are sets of partial constituency trees that are licensed by

rules of the grammar. As a result, parsing schemas allow compilation of schemas to executable parsers and also permit formal reasoning about properties of the parser directly. Unlike many deductive systems used to define parsers, there is no one-to-one relationship between a parsing schema and algorithm. In later work, Alonso Pardo et al. (1999) showed parsing schemas can be used to define parsers for other grammar formalisms such as tree-adjoining grammars.

This book, which is an extended version of Carlos Gómez Rodríguez's Ph.D. thesis work, extends the theory and practice of parsing schemas in several directions. There are three major parts to this book:

Compiling and executing parsing schemas. The first part of the book provides a language syntax that can be used to precisely specify parsing schemas and a compiler for this domain-specific language. The syntax is shown in the CYK example above. The full specification also includes interpretations for indices such as i and $i + 1$ as word positions, A, B, C as grammar symbols, and how deduction steps and goals are converted into parser code. The implementation details, including static analysis of the schema and the Java code generation, are described well and in sufficient detail. The source code of the compiler for parsing schemas is available for download at `www.grupocole.org/software/COMPAS`. (The code is typical research software—it takes some effort to use it, but once you do, you can play with compiling and running most of the schemas in the book.) This part of the book contains experiments on comparing many different parsing schemas for each of these formalisms. The comparison is done using hand-written grammars with feature structures (the parsers include feature unification) and evaluated on test-suite data rather than on modern Treebank grammars and data from newswire and other "real-world" data. Another issue is that the comparison does not include the GHR parser (Graham, Harrison, and Ruzzo 1980), which may impact the comparison between CYK and Earley parsers. Also, interesting synthetic-data experiments are presented that compare tree-adjoining parsers with context-free parsers; it is not clear whether these results extend to natural language corpora. With regard to implementation of schemas, the focus is mainly on agenda-based implementation of deductive steps rather than, say, the use of (pushdown) transducers to produce parse trees.

Error-repair parsers. The second part of the book focuses on error-repair in parsing (using parsing schemas, of course). Such an approach tries to deal with limited coverage of the grammar by performing insertions, deletions, or substitutions on the input string. This makes a lot of sense in programming language parsers, but for natural languages it makes little sense to transform the input because the grammar has poor coverage. It is trivial to add (weighted) glue rules that accept any input string, or a finite-state acceptor of strings can be used as a back-off grammar to improve coverage. Speech repair and other such cases are typically handled using appropriate augmentations of the underlying grammar combined with grammar-driven edits (Charniak and Johnson 2001). Despite this, error-repair is a good use-case for parsing schemas. Gómez Rodríguez can show that some existing error-repair parsers are in fact provably correct, and also a generic recipe can be given that converts any given parser schema into an error-repair parser schema. This is an instructive use of parsing schema transformations, because it is easy to show that the changes preserve correctness.

Parsing schemas for dependency parsers. This third part of the book has the potential to be the most popular. There is increased interest in multilingual dependency parsing, and there are a large number of different dependency parsing algorithms. Parsing schemas

allow a concise description of many different parsing algorithms, and Gómez Rodríguez provides many parsing schemas corresponding to popular dependency parsing algorithms; there are too many to list here, but he provides schemas for no less than ten dependency parsers, including some that recover non-projective dependencies. He also provides explicit relationships between schemas for these varied parsers, such as item refinement (an item deduced in one parser is broken up as multiple items in another parser) or step refinement (a deduction step in one parser can be emulated by a sequence of steps in another parser). It is also useful that these relationships are transitive and reflexive. However, it is in describing dependency parsing that the biggest weakness of parsing schemas is exposed and its potential role as an universal language for the parsing community runs into trouble. Non-constructive aspects of parsing cannot be represented with a schema, because that violates the semantics of deductive steps. For instance, in a parser that computes the dependency tree by using the minimum spanning tree (MST) algorithm (McDonald et al. 2005), there is a step that eliminates cycles in the graph. This step is not constructive and therefore the MST parser cannot be represented as a schema. Parsing schemas are generally grammar-driven and often parsers are written without any finite underlying grammar, which makes tree building harder to describe concisely.

The discussion of related work touches on the use of Prolog for parsing schemas (Shieber, Schabes, and Pereira 1995), Datalog for specifying parsers (McAllester 2002; Liu and Stoller 2003), the DyALog system (Villemonte de la Clergerie 2005), and Dyna (Eisner, Goldlust, and Smith 2005). It is true that Dyna is quite powerful because it is a full general-purpose declarative programming language, but for that reason it offers an attractive alternative to parsing schemas. On the other hand, schemas do allow formal reasoning about parsers that may be more fine-grained than is possible in Dyna. Surprisingly, work on semiring parsing (Goodman 1998, 1999) is not mentioned. The use of probabilities or weights is generally ignored in this book, even though it enables interesting methods for speeding up parsers such as coarse to fine parsing (Goodman 1997) or generalized A$*$ search for parsing (Pauls and Klein 2009). While there is more than enough content in this book, it does not cover the use of parsing schemas in machine translation. In particular, formal properties of schemas might make it easier to describe and implement the integration of language models into parsing algorithms for synchronous context-free grammars (Chiang 2007). Schemas might have much to offer with respect to proving correctness in machine translation decoders.

The potential reader for this book is likely to be a parsing enthusiast curious about the power of schemas to represent parsing algorithms succinctly and to prove them correct. They might also be interested in showing relationships between their novel parsing schemas and other well-known parsers, or showing how extensions to existing parsers are well justified. Dependency parsing enthusiasts who want to wrap their head around the many different parsing algorithms out there might also be interested in the concise description of such parsers.

References

Alonso Pardo, Miguel A., David Cabrero Souto, Eric de la Clergerie, and Manuel Vilares Ferro. 1999. Tabular algorithms for TAG parsing. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 150–157, Bergen.

Charniak, Eugene and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Eisner, Jason, Eric Goldlust, and Noah A. Smith. 2005. Compiling comp ling: Weighted dynamic programming and the Dyna language. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 281–290, Vancouver, Canada.

Goodman, Joshua. 1997. Global thresholding and multiple-pass parsing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing: EMNLP-1997*, pages 11–25, Providence, RI.

Goodman, Joshua. 1998. *Parsing inside-out*. Ph.D. thesis, Harvard University, Cambridge, MA.

Goodman, Joshua. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Graham, Susan L., Michael Harrison, and Walter L. Ruzzo. 1980. An improved context-free recognizer. *ACM Transactions on Programming Languages and Systems*, 2(3):415–462.

Huang, Liang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Cambridge, MA.

Huang, Liang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Uppsala.

Kasami, Tadao. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

Liu, Yanhong A. and Scott D. Stoller. 2003. From Datalog rules to efficient programs with time and space guarantees. In *Proceedings of the 5th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pages 172–183, Uppsala, Sweden.

McAllester, David. 2002. On the complexity analysis of static analyses. *Journal of the Association for Computing Machinery*, 49(4):512–537.

McDonald, Ryan T., Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technologies and Empirical Methods in Natural Language Processing Conference*, pages 523–530, Vancouver, Canada.

Pauls, Adam and Dan Klein. 2009. Hierarchical search for parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 557–565, Boulder, CO.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36.

Sikkel, Klaas. 1997. *Parsing Schemata: A Framework for Specification and Analysis of Parsing Algorithms*. Springer, Berlin.

Villemonte de la Clergerie, Éric. 2005. DyALog: a Tabular Logic Programming based environment for NLP. *Proceedings of the 2nd International Workshop on Constraint Solving and Language Processing*, Barcelona, Spain.

Younger, Daniel H. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208.

Anoop Sarkar is an Associate Professor at Simon Fraser University, Burnaby, BC, Canada, where he co-directs the Natural Language Laboratory (`natlang.cs.sfu.ca`). His research is focused on statistical parsing and machine translation. His interests also include formal language theory and stochastic grammars, in particular tree automata and tree-adjoining grammars. His e-mail address is `anoop@cs.sfu.ca`.