Book Reviews

Finite State Morphology

Kenneth R. Beesley and Lauri Karttunen
(Xerox Research Centre Europe and Palo Alto Research Center)

Stanford, CA: CSLI Publications (CSLI studies in computational linguistics, edited by Ann Copestake) (distributed by the University of Chicago Press), 2003, xviii+505 pp and CD-ROM; hardbound, ISBN 1-57586-433-9, $85.00, £59.50; paperbound, ISBN 1-57586-434-7, $40.00, £28.00

Reviewed by
Shuly Wintner
University of Haifa

Finite-state technology (FST) is a general term for the use of finite-state automata and transducers in computational linguistics and natural language processing (NLP). FST is very versatile, having been used very successfully for describing the phonology, orthography, and morphology of a large number of languages, as well as for solving practical problems such as morphological analysis and generation, language modeling for speech processors, shallow parsing, segmentation, and named-entity recognition. This technology is now very mature: Ever since it was observed by Johnson (1972) that the kind of phonological rules that are used by linguists denote, in fact, regular relations, and especially since the pioneering work of Koskenniemi (1983) and Kaplan and Kay (1994), much work has been invested in improving algorithms for finite-state networks and creating more regular-expression-like operators that can be compiled into finite-state networks.

It is therefore surprising that no textbook covering this technology in detail has previously been published: Beesley and Karttunen's *Finite State Morphology* is, to the best of my knowledge, the first textbook that is dedicated to this subject. Even general NLP textbooks spend relatively little space on FST (two pages in the case of Allen [1995], two sections out of twenty-five in the case of Jurafsky and Martin [2000]). The book is dedicated to a particular implementation, which comes with two regular expression languages, XFST and LEXC, and with compilers that can translate the expressions to extremely space- and time-efficient networks. Both systems were designed and implemented by Xerox and are provided (with compilations for Solaris, Linux, Windows, and Mac OS X) on a CD that accompanies the book.

The audience for the book is mainly linguists, and not necessarily computational linguists or computer scientists. The authors deliberately avoid mathematical definitions and specifications of algorithms, let alone proofs, in the text. However, mathematical correctness is not compromised, although it is doubtful whether readers with limited formal background would be able to appreciate it. Readers who *are* interested in the mathematics and in the computational aspects of the implementation will be left unsatisfied. Not only are they missing in the text, there are relatively few

references to such works (for example, there is hardly any reference to Mohri's [1997] works on sequential transducers, to Daciuk and others' works on incremental construction of lexicons [Daciuk et al. 2000], or to van Noord and Gerdemann's [2001a] work on transducers with predicates). There is also no mention of weighted finite-state networks and their uses in natural language processing.

For linguists, however, the book is full of useful advice. Not only does it provide a very gentle introduction to the field (chapter 1) and an extremely detailed and very well exemplified description of finite-state networks in general (chapter 2) and of the Xerox tools in particular (XFST in chapter 3 and LEXC in chapter 4), but it also provides invaluable insight into the process of developing large-scale finite-state networks, from the design and planning phase through maintenance, testing, and debugging (chapters 5 and 6). The core of the book consists of chapters 3 and 4, in which the authors describe in great detail the two main tools. Each and every operator is defined, explained, and demonstrated, usually with very illuminating linguistically motivated examples. The discussion is accompanied by useful exercises, many of which are solved in an appendix.

Programming with regular expressions is very different from programming in conventional languages (procedural, functional, or logic). The book provides an excellent exposition of the material, emphasizing not only the syntax and semantics of the two languages, XFST and LEXC, but also tips and tricks for clear and efficient network construction and common pitfalls to avoid. The detailed examples provide real-life morphological problems and the correct way to solve them. Thus, a considerable subset of Esperanto morphotactics is actually covered by examples in chapter 4. Examples are also drawn from Spanish, Portuguese, Irish, Arabic, Malay, and a variety of other natural languages.

The remainder of the book is dedicated to more marginal issues: flag diacritics, a special mechanism for tagging finite-state networks which is reminiscent of ATNs, are discussed in chapter 7, whereas provisions for nonconcatenative morphology, and in particular the compile-replace algorithm, are described in chapter 8. Some Xerox utilities are then listed in chapter 9, which closes the book.

Pedagogically, this is an extraordinary book. Its organization is excellent, no concept is used without being defined and exemplified, and key notions are repeated over and over again. Most chapters start with an introduction that summarizes the previous material and motivates the discussion and end in a summary. The authors' love for language, and in particular morphology, is evident everywhere, especially in the examples of "the mythical Bambona language" or "the fictional Monish language." It makes the book very enjoyable reading. It is evident that the book builds on many years of extensive experience with the technology in general and the Xerox tools in particular, and extensive experience *teaching* these issues; the authors do not hesitate to share this vast experience with their readers. For teachers of introductory NLP classes, as well as more advanced courses on FST, this book is a gold mine.

Who should buy this book? If you are a linguist who is planning to do some work with finite-state technology, then this book is a must. If you are not sure whether FST is for you, this book will most likely convince you that it is. If you are already working with some finite-state toolbox (such as the *FSM tools* from AT&T [Mohri, Pereira, and Riley 1998] or van Noord and Gerdemann's [2001b] *FSA utils*), then this book will provide insight into the vast possibilities that the technology offers and will help you place your own work in context. Specifically, if you already work with LEXC or XFST, this is the bible of the applications. However, if you are interested in the mathematics of finite-state networks or in the computational aspects of their implementations, you are probably better off with a book such as Roche and Schabes (1997).

A minor note: The book is extremely well-written, its language is fluent and lucid, and hard as I tried, I could not find a single error or typo. However, the repeated references to Xerox, the Xerox linguists, and the Xerox developers of the technology described in the book are exaggerated. The book would have been more fun to read without them.

References

Allen, James. 1995. *Natural Language Understanding*, second edition. Benjamin/Cummings, Redwood City, CA.

Daciuk, Jan, Stoyan Mihov, Bruce W. Watson, and Richard E. Watson. 2000. Incremental construction of minimal acyclic finite-state automata. *Computational Linguistics*, 26(1): 3–16.

Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. Mouton, The Hague.

Jurafsky, Daniel and James H. Martin. 2000. *Speech and Language Processing*. Prentice Hall, Upper Saddle River, NJ.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Department of General Linguistics, University of Helsinki, Helsinki, Finland.

Mohri, Mehryar. 1997. On the use of sequential transducers in natural language processing. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing: Language, Speech and Communication.* MIT Press, Cambridge, MA, pages 355–381.

Mohri, Mehryar, Fernando Pereira, and Michael Riley. 1998. *A Rational Design for a Weighted Finite-State Transducer Library* (volume 1436 in Lecture Notes in Computer Science). Springer, Berlin/New York.

Roche, Emmanuel and Yves Schabes, editors. 1997. *Finite-State Language Processing*: *Language, Speech and Communication.* MIT Press, Cambridge, MA.

van Noord, Gertjan and Dale Gerdemann. 2001a. Finite state transducers with predicates and identity. *Grammars*, 4(3): 263–286.

van Noord, Gertjan and Dale Gerdemann. 2001b. An extendible regular expression compiler for finite-state approaches in natural language processing. In O. Boldt and H. Jürgensen, editors, *Automata Implementation* (volume 2214 in Lecture Notes in Computer Science). Springer, Berlin/New York, pages 122–139.

Shuly Wintner is a lecturer at the University of Haifa. His research interests involve various areas of computational linguistics, and in particular, the application of methods and techniques from computer science to the study of linguistic formalisms. He is also doing research on the morphology and syntax of Semitic languages, especially Hebrew. Wintner's address is Department of Computer Science, University of Haifa, 31905 Haifa, Israel; e-mail: shuly@cs.haifa.ac.il.