Kenneth R. Beesley and Lauri Karttunen

Finite State Morphology

CSLI Publications, Stanford, California, 2003
(CSLI Studies in Computational Linguistics,
xviii+510 pp and CD-ROM, ISBN 1-57586-434-7)
Center for the Study of Language and Information,
Leland Stanford Junior University

Reviewed by
Otakar Smrž

Content

The book is a reference guide to the finite-state computational tools developed by Xerox Corporation in the past decades, and an introduction to the more general techniques useful for modeling of morphology in mathematical and computer linguistics.

The volume of more than five hundred pages is divided into two opening chapters on the history of the book and the preliminaries for using it, nine chapters on the topic proper, two appendices of additional quizzes and solutions to exercises, a list of references and an index. The book includes a CD-ROM with the software, and is kept up-to-date through the `http://www.fsmbook.com/` website.

The first chapter, A Gentle Introduction, conveys the core concepts of the book in terms of analogies and examples, purposely avoiding rigorous language and demonstrating the terminology in a very intuitive way. Characteristics and applications of finite-state networks are given like this along with a background in set theory.

A Systematic Introduction, the next chapter, then proceeds with more formal definitions of *language*, *relation*, *finite-state network*, or *regular expression*. It is explained that languages and relations can be denoted by regular expressions and encoded by finite-state networks, i.e. automata and transducers, respectively, and the compilation of some elementary regular expressions into the networks is outlined. A wide range of regular-expression operators is introduced. Most notably the nonstandard ones, such as *ignoring*, *crossproduct*, *projection*, *composition*, *restriction* or various types of *replacement*. In separate sections and paragraphs, several non-trivial considerations on the properties of finite-state networks are presented and discussed.

The third chapter, The `xfst` Interface, is an extensive tutorial to the general-purpose interpreter of commands or scripts designed for creating and manipulating finite-state networks. It tells the syntax for implementing regular expressions, defining their possibly multicharacter alphabets, compiling them and storing them in variables or files, etc. There is a special mechanism for stacking the transducers on top of each other, and a suite of commands for testing their performance, viewing their properties, and, of course, applying the networks on the languages being modeled.

Chapter four, The `lexc` Language, offers an alternative formalism to defining finite-state networks. The language is purely declarative and, in principle, encodes right linear grammars (Rozenberg and Salomaa, 1997), which can be compiled to equivalent finite-state networks. The term *grammar* is actually not defined explicitly in this context. We could describe the `lexc` concepts of *continuation classes* and *lexicons* as *nonterminals* and the corresponding *rewrite rules*, once drawing a parallel with the notions of the theory of formal languages.

Planning and Managing Finite-State Projects then summarizes some important tips for effective organization of work involved in building finite-state linguistic models. The chapter covers not only software engineering issues like version control and modularity requirements, but rather provides hints for implementing different spellings, lexicon restrictions, etc. without the need to modify copies of the original grammars.

The sixth chapter, Testing and Debugging, concerns itself with automated processing of language resources and their exploitation for revealing the errors or misfunctions of the models. The `tokenize` and `lookup` executables are mentioned for the first time, being the runtime utilities applying the finite--state networks in practical data-stream jobs.

Flag Diacritics, introduced in the seventh chapter, are an extension to the finite-state calculus. The notation controls the computation in a network by requiring compatibility of the feature-setting and feature-unification operations imposed on its transitions. Implementing long-distance constraints in the pure finite-state paradigm expands the networks in size. Flag Diacritics allow to enforce the desired dependencies while keeping the grammars and the networks simple.

Non-Concatenative Morphotactics of chapter eight proposes lexicon-efficient modeling of the problematic morphological phenomena like *reduplication* or *interdigitation*. The trick is that both the upper and the lower language of a network can be interpreted as a regular expression and re-compiled into a more complex finite-state network. Running the `compile-replace` command in `xfst` is similar to generating and evaluating some code in other programming languages.

The closing chapter, Finite-State Linguistic Applications, returns in greater detail to the `tokenize` and `lookup` utilities and suggests how to build more sophisticated tokenizers, morphological guessers, spelling checkers and correctors, shallow parsers and the like by performing virtual composition of finite-state networks and designing miscellaneous lookup strategies.

Critique

Unfortunately, the quality of the information given throughout the book is uneven and does not seem to fit properly to any target audience. The style of a non-trivial and well-written popularization, especially in the first two chapters and some of the final ones, is degraded by evident, yet repeated comments on the methodology of using the concrete interfaces for finite-state modeling, by quite lengthy examples and, occasionally, too strong formulations or omissions irrespective of the state of the art in information technology nowadays.

The results of the theoretical research in the finite-state calculus should be clearly distinguished from their application in computational morphology, as well as from the formalisms and implementations constituting the working environments. While the former is highly appreciated and inspiring, the very last has, in our view, been already overcome and is not worth trying to follow unless the programming and user interfaces improve in flexibility and the language models become reusable. The authors shortly advertise XeLDA, the runtime environment for advanced research and commercial applications, but it is not licensed with the book, nor described therein. The sites `http://www.mkms.xerox.com/` or `http://www.temis-group.com/temis/attachments/factsheets/xelda_en.pdf` promise, still, that there might be some APIs ready for the `xfst` core algorithms, and that modularity and the freedom of data formats are supported as one would expect.

It is to clarify that finite-state modeling of morphology in Xerox has been remarkably successful, and that significant work has been done even for Czech and many non-European or challenging languages such as Turkish, Finnish, Hungarian, Malay, Basque or Esperanto, and Arabic (Beesley, 2001), `http://www.arabic-morphology.com/`. However, the development of the systems and, more seriously, their maintenance and incorporation into other applications, was experienced on our side as inadequately difficult and clumsy due to the non-openness of the underlying architecture.

Coming back to the publication, the CD-ROM could have comprised the scripts and examples presented in the text, and not only the 8 MB of platform-diversified executables. Such a favor would have been much more instructive, giving the users the great chance to experiment with the tools and notations, and the book itself would have become clearer and nicer. We also had an opportunity to read the pre-publication review copy of this book, and regret a little that the tenth chapter on complexity and finite-state solutions to constraint problems did not survive into the ultimate version of the work.

Let us conclude that the book does contribute to the knowledge in the field, represented e.g. by (Rozenberg and Salomaa, 1997), (Sproat, 1992), (Kornai, 1999). Bringing to light again that regular relations are not closed on intersection and have even other computational limitations compared to regular languages, offering the original treatment of Flag Diacritics and thus justifying the ideas in (Mokrý and Smrž, 2002), or extending on the gist of non-concatenative finite-state modeling next to (Kiraz, 2001), all these will be referred to by us as the book's highlights.

References

Beesley, Kenneth R. 2001. Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001. In *EACL 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pages 1–8, Toulouse, France, July 2001.

Kiraz, George Anton. 2001. *Computational Nonlinear Morphology with Emphasis on Semitic Languages*. Studies in Natural Language Processing. Cambridge University Press.

Kornai, András, editor. 1999. *Extended Finite State Models of Language*. Studies in Natural Language Processing. Cambridge University Press.

Mokrý, Karel and Otakar Smrž. 2002. External Tools Not Only for ArabTeX Documents. In *Proceedings of the International Symposium on Processing of Arabic*, pages 161–165, Manouba, Tunisia, April 2002.

Rozenberg, Grzegorz and Arto Salomaa, editors. 1997. *Handbook of Formal Languages*. Springer.

Sproat, Richard. 1992. *Morphology and Computation*. ACL–MIT Press Series in Natural Language Processing. MIT Press.