Book Reviews

PC-KIMMO: A Two-Level Processor for Morphological Analysis

Evan L. Antworth
(Summer Institute of Linguistics)

Dallas: Summer Institute of Linguistics, 1990, xii + 273 pp. and disks for IBM PC or Macintosh[1] (Occasional Publications in Academic Computing, Number 16) Paperbound, ISBN 0-88312-639-7, $23.00

Reviewed by
Richard Sproat
AT&T Bell Laboratories

This package is a faithful implementation of Koskenniemi's two-level morphology, KIMMO (Koskenniemi 1983). As with previous tools from the Summer Institute of Linguistics (e.g., Weber, Black, and McConnel 1988), the package is primarily aimed at field linguists with little computational expertise, but for whom morphological analysis tools are useful. The book that accompanies the software is extremely good: among other things, it incorporates what is in my opinion the best available introduction to the theory and practice of two-level morphology. The package is therefore not only useful to the audience for which it is primarily intended: it will also be of use to those wishing to learn something about computational morphology, and would be an excellent teaching aid for a computational linguistics course.

The book is divided into seven chapters and three appendices. Chapter 1 traces the history of computational morphology. As is typical of such overviews, the scope is somewhat narrow, discussing only a few Finnish projects (besides the work of Koskenniemi and his apostles), and work done at the Summer Institute of Linguistics. That criticism aside, the rest of the chapter is an accurate overview of the positive and negative aspects of KIMMO. On the negative side, Antworth cites the well-known fact that KIMMO cannot elegantly handle nonconcatenative morphology such as infixation or reduplication. He also notes limitations more particular to the PC-KIMMO implementation, such as the lack of a rule compiler and the lack of a sensible (i.e., non–finite-state) model of morphotactics; both of these issues have received treatment within the two-level school (Dalrymple et al. 1987, Bear 1986, inter alia), though with respect to rule compilation, Antworth is justified in noting (p. 11) that "it remains to be seen if such a project is feasible for small computers."

Chapter 2 is an introduction to PC-KIMMO, and gives a sample session of its use. Chapters 3 through 5 describe in detail how finite state transducers (FSTs) work,

1 The software runs on IBM PCs and compatibles running MS-DOS or PC-DOS version 2.0 or higher; minimum requirements are 256K memory (will use up to 640K); executable is about 97.5K. Versions are available for the Macintosh, and the system has been compiled and tested under Unix System V and 4.2 BSD Unix.

how to write two-level rules, how to hand-compile two-level rules into state transition tables, how to develop a lexicon, and how to test and debug a two-level description. The discussion of FSTs and two-level rules in Chapter 3 is particularly good, being both extensive and clear. One important point that is stressed is that to code two-level rules as FSTs, one needs to think in terms not of what the rules accept, but rather of what they reject (what Koskenniemi (p. 40) calls REJECTION SETS). As Antworth notes (p. 45) "this is one of the most difficult barriers to overcome in learning to write two-level rules and state tables." The discussion of the technical issues in these chapters is accurate. The only point I find myself taking issue with is the statement (pp. 28–29) that "because two-level rules [as opposed to generative phonological rules] express a correspondence rather than rewrite symbols, they apply in parallel rather than sequentially," and "thus no intermediate levels of representation [like those of generative phonology] are created." The notions 'express a correspondence' and 'apply in parallel' are not logically related. One can surely have a system in which rules 'express a correspondence' between intermediate levels of representation, and traditional generative phonology is formally an instance of this. Furthermore, in their FST implementation, two-level rules work by consulting an input character and *transducing* it to some character (or null) in the output (see also Barton et al. 1987, p. 147). Thus, the difference between the 'procedural' rewriting view of generative phonology and the 'declarative' relational view of two-level phonology has more to do with the rhetorical style of the practitioners of the two fields than with any deep formal differences. The only difference is that generative descriptions allow intermediate levels of representation, whereas two-level descriptions do not.

Chapter 6 contains some illustrative PC-KIMMO descriptions of various natural phenomena, including Turkish vowel harmony, French consonant deletion and Tagalog infixation and reduplication: in addition, Appendix A contains an English morphology fragment, and Appendix B contains some further illustrations of how PC-KIMMO can be used, including an amusing implementation of a logic puzzle. Many of the descriptions in Chapter 6 and the appendices, as well as material not discussed in the text, are distributed along with the software. The treatment of infixation and reduplication is particularly noteworthy, since it illustrates two points rather nicely:

1. such processes *can* be handled in the KIMMO framework, but
2. it would be painful to implement a system of any degree of complexity.

Tagalog infixation and reduplication are both simple instances of their respective species, yet the machines are somewhat large for hand-coding purposes. In the description of more complicated nonconcatenative processes, the machines become very large and impractical to hand code; for Warlpiri reduplication, one would require an FST with many thousands of states.

The PC-KIMMO software is easy to use for several reasons. First of all, the user's manual (Chapter 7) is clearly written. Second, the error messages are usefully specific. Third, a number of useful debugging and development facilities are available, including: the ability to call up an editor of one's choice from within PC-KIMMO; a tracing facility with several available levels of verbosity; a file comparison mode allowing one to test the generator and recognizer on pairs of lexical and surface forms taken from a file, thus facilitating an extensive testing of one's description. For those wishing to develop other software that includes morphological processing capabilities, a PC-KIMMO source library is included in the release. The functions are fully documented in Appendix C, and also included is some sample code illustrating their

use. The code runs tolerably fast: in the English fragment provided in the release, recognition of a moderately long word form takes about 0.6 seconds on an AT&T 6300 Plus (and is about 17 times faster on a SPARCstation 1). As noted above, there are some things, notably a rule compiler, that one might like ultimately to have in such a system. Nonetheless, PC-KIMMO is already a very useful and accessible tool.

References

Barton, G. Edward; Berwick, Robert; and Ristad, Eric (1987). *Computational complexity and natural language.* Cambridge MA: The MIT Press.

Bear, John (1986). "A morphological recognizer with syntactic and phonological rules." *Proceedings, International Conference on Computational Linguistics (COLING-86)*, Bonn, 272–276.

Dalrymple, Mary; Kaplan, Ronald; Karttunen, Lauri; and Koskenniemi, Kimmo (1987). "Tools for morphological analysis." Technical Report CSLI-87-108, Center for the Study of Language and Information, Stanford.

Koskenniemi, Kimmo (1983). *Two-level morphology: A general computational model for word-form recognition and production.* Ph.D. thesis, University of Helsinki, Helsinki.

Weber, D.; Black, H. A.; and McConnel, S. (1988). *AMPLE: A tool for exploring morphology* (Occasional publications in academic computing). Dallas: Summer Institute of Linguistics.

Richard Sproat is a member of the technical staff in the linguistics research department at AT&T Bell Laboratories. He holds a doctorate in Linguistics from the Massachusetts Institute of Technology. His research interests include morphological theory, and he has done work in various aspects of computational morphology. His address is: AT&T Bell Laboratories, 600 Mountain Avenue (Room 2d-451), Murray Hill, NJ 07974. E-mail: rws@research.att.com

Two-level morphology: consequences. â— Morphological grammars for many languages using Koskenniemiâ€™s approach. â—‹ E.g., PC-KIMMO (Antworth, 1990) developed for use by field linguists. â— Stimulated debate on whether deep representations were required in phonology. â—‹ E.g., declarative â€œone-level phonologyâ€ (Bird & Ellison, 1994).Â 1990. PC-KIMMO: A Two-Level Processor for Morphological Analysis. SIL Publications, Dallas. â— Mark Aronoff. 1976. Word Formation in Generative Grammar.Â 1987. â€œReview of text-to-speech conversion for English.â€ Journal of the Acoustical Society of America 82, 737-793. â— Kimmo Koskenniemi. 1983. Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production. PhD Thesis, University of Helsinki.