13

Morphology and the Harappan Gods

RICHARD SPROAT AND STEVE FARMER

13.1 Introduction

Kimmo Koskenniemi has done work in a variety of areas having to do with the computational modeling of language, including computational syntax, information retrieval and, most famously, computational morphology. It is this latter area, and one other perhaps less well-known one, that are the topic of this chapter.

Koskenniemi's thesis work on the computational modeling of Finnish morphology (Koskenniemi, 1983) is certainly the best-known work in the field of computational morphology, and it has inspired a wealth of derivative work, including practical working morphological analyzers for a wide variety of languages.

One of his lesser known contributions is in the area of decipherment, namely his collaboration with the Finnish Indologist Asko Parpola on the computational analysis of the inscriptions of the Indus Valley.

In this chapter we will review these two contributions and their importance for their respective fields. Note that the first author of this paper may possibly be the only other person in the world who, like Koskenniemi, has done work on these two topics. The second author is the first author's collaborator on the Indus Valley work.

13.2 Koskenniemi's Contributions in Morphology

Koskenniemi's development of Two-Level Morphology can be thought of as a fortuitous accident of history. It had been known since C. Douglas Johnson's PhD thesis 1972 that "context-sensitive" rewrite rules of the kind that

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

had become familiar in generative phonology described regular relations and could thus be implemented using finite-state transducers (FSTs). By the late 1970's Ron Kaplan and Martin Kay at Xerox PARC were developing algorithms for the automatic compilation of FSTs from rewrite rules in a format that would be familiar to linguists, namely:

$$\phi \rightarrow \psi / \lambda \_ \rho \tag{13.1}$$

Here, ϕ, ψ, λ and ρ could be arbitrary regular expressions. Furthermore, since regular relations are closed under composition, this meant that one could write a series of ordered rules of the kind found in SPE (Chomsky and Halle, 1968), compile each of the rules into a transducer and then compose the entire series of rules together to form a single transducer representing the entire rule system. Kaplan and Kay finally published their algorithm many years later (Kaplan and Kay, 1994), and there has been subsequent work on a simpler and more efficient algorithm in Mohri and Sproat (1996).

But in the late 1970's and early 1980's there was just one problem: computers were simply not fast enough, nor did they have enough memory to compile rule systems of any serious complexity. Indeed complex rule systems of several tens of rules over a reasonable-sized alphabet (say 100 symbols) can easily produce FST's with several hundred thousand states with a similar number of arcs, with a total memory footprint of several megabytes. While any PC today could easily handle this, this was simply not viable around 1980.[1]

13.2.1 The two-level morphological system

Koskenniemi therefore proposed an alternative, one that still used transducers but constructed them and used them in a different way. First of all, he eschewed rule compilation entirely, instead constructing his transducers by hand. This is not quite as bad as it seems, since he proposed various ergonomically reasonable devices, such as the use of a "wildcard" ('=' in his notation) that would match any character not already mentioned: thus for any state, one could specify transitions to other states on designated symbol pairs, and have a default transition on '=:=' if none of the other specifications matched. This allowed the FSTs in Koskenniemi's description to be quite compact.

Second, rather than deal with rule composition, he came up with a novel alternative: the FSTs would run in parallel, each of them reading characters from the surface tape (the form of the word that appears in text) and the lexical tape (the form of the word that is entered in the lexicon, along with its morphosyntactic features). This presents a theoretical problem though, because a system of this kind is implementing *intersection* of FSTs and hence regular

[1] Recall Bill Gates' 1981 statement that "640k ought to be enough for anybody."

relations, whereas it is known that regular relations are not generally closed under intersection (Kaplan and Kay, 1994); however so long as the number of insertions or deletions is bounded, it can be shown that regular relations are closed even under intersection (Roark and Sproat, 2006), and in effect this is what Koskenniemi's system is doing when it constrains the transducers from getting too out of sync.

Koskenniemi's implementation of the lexical entries themselves, as well as affixes was less of an innovation. For the lexicons, he used the idea of letter *tries*, from Knuth (1973). To handle morphological decomposition he used the notion of *continuation lexicon* where a lexical entry would be annotated with information on what other lexical entries (usually affixes) could follow it. But this is just an implementation of a finite-state grammar and in fact Koskenniemi's trie-plus-continuation-lexicon approach is formally equivalent to representing the lexicons as finite-state acceptors (FSAs).

In Koskenniemi's original formulation, the input (surface) word would be matched against the lexicon by starting at a root lexicon and then matching the characters of the input against the characters in the lexicon trie, modulated by the parallel phonological transducers, which Koskenniemi picturesquely describes as viewing the lexicon through a slightly distorting lens. A present day two-level system would, of course, implement the following set of finite-state operations, where I is the input word, R_i are the rule transducers, and L is a lexical FSA:

$$I \circ \bigcap_i (R_i) \circ L \tag{13.2}$$

13.2.2 Two-Level Rules

The other innovation of Koskenniemi's approach was his formalization of two-level rewrite rules; again, he did not provide a compiler for these rules, but the rules served to specify the semantics underlying the transducers that he built by hand. All rules in his system followed a template in that they were all of the following form:

CorrespondencePair **operator** LeftContext _ RightContext

That is, the rules specified conditions for the occurrence of a correspondence pair — a pairing of a lexical and a surface symbol (one of which might be empty), modeling deletion or insertion — in a given left or right context. The contexts could be regular expressions, but the correspondence pair was a single pair of symbols, and thus was not as general as the $\phi \rightarrow \psi$ formulation from Kaplan and Kay (1994).

Koskenniemi's rules came in four flavors, determined by the particular **operator** used. These were:

Exclusion rule	a:b /⇐ LC _ RC
Context restriction rule	a:b ⇒ LC _ RC
Surface coercion rule	a:b ⇐ LC _ RC
Composite rule	a:b ⇔ LC _ RC

The interpretation of these was as follows:

- **Exclusion rule**: *a* cannot be realized as *b* in the stated context.
- **Context restriction rule**: *a* can only be realized as *b* in the stated context (i.e. nowhere else)
- **Surface coercion rule**: *a* must be realized as *b* in the stated context.
- **Composite rule**: *a* is realized as *b* obligatorily and only in the stated context.

In many ways the semantics of Koskenniemi's rules was better defined than the ones that had previously been used in generative phonology. For one thing, each rule type specified a direct relation between the underlying and surface forms, something that was not possible within generative phonology due to the arbitrary number of ordered rewrite rules: in general, in generative phonology there was no way to know how a given lexical form would surface, short of applying all rules in the specified order and seeing what the outcome was. Koskenniemi's rules, in contrast, specified the relation directly.

Ignoring for the moment that traditional generative phonological rules were not two-level, one can ask which of Koskenniemi's rules correspond to the rule types (basically just obligatory or optional rewrite rules) of generative phonology. In fact only the **surface coercion rule** has a direct counterpart: it corresponds pretty directly to an obligatory rewrite rule. All the other two-level rule types depend upon global knowledge of the system. Thus the **context restriction rule** is equivalent to a situation in a traditional generative account where there is but one optional rule that changes *a* into *b*; but note that this is a property of the system, not of a specific rule. The **composite rule**, which is just a combination of **context restriction** and **surface coercion** is similar, but in this case the unique rule changing *a* into *b* is obligatory. Note that since one could write, say, a **context restriction** rule that relates *a* to *b* in one environment, and then also write another **context restriction** rule that allows *a* to become *b* in another environment, it is perfectly possible in Koskenniemi's system to write an inconsistent grammar. A lot of the work in designing later two-level systems involved writing debuggers that would catch these kinds of conflicts. Finally, the **exclusion rule** is again global in nature: it is equivalent to the situation in a traditional generative grammar where there is no rule that relates *a* to *b* in the specified environment.

But really, Koskenniemi's rules can best be thought of as involving constraints on correspondence pairs. Constraints were virtually non-existent as a device in early generative phonology, but have since become quite popular

in various theories of phonology including Declarative Phonology (Coleman, 1992), One-Level Phonology (Bird and Ellison, 1994) and Optimality Theory (Prince and Smolensky, 1993).

13.2.3 Koskenniemi's impact on computational morphology

Koskenniemi's two-level morphology was remarkable in another way: in the early 1980's most computational linguistic systems were toys. This included parsers, which were usually fairly restricted in the kinds of sentences they could handle; dialog systems, which only worked in very limited domains; and models of language acquisition, which were only designed to learn simple grammatical constraints. In contrast, Koskenniemi's implementation of Finnish morphology was quite real in that it handled a large portion of inflected words that one found in real Finnish text. To some extent this reflects the fact that it is easier to get a quite complete coverage of morphology in any language than it is to have a similar coverage of syntax, let alone dialog. But it also reflects Koskenniemi's own decision to develop a full-fledged system, rather than present a mere "proof of concept" of his ideas.

While two-level morphology was originally motivated by the difficulties, at the time, with Kaplan and Kay's approach to cascaded rewrite rules, the model quickly took on a life of its own. Koskenniemi took it to be a substantive theoretical claim that only two levels of analysis were necessary, a claim that was fairly radical in its day (at least in contrast to generative phonology), but which has since been superseded by claims that only one-level is needed (e.g. Bird and Ellison, 1994).

Nevertheless, practical considerations of developing morphological analyzers have led people to not rely wholly on the two-level assumption. Since transducers can be combined both by composition (under which they are always closed) and by intersection (under which they are closed under certain conditions) combinations of these two operations may be used in any given system; see, e.g., Karttunen et al. (1992). Indeed, one of the beauties of finite-state techniques is that the calculus of the combination of regular languages and relations is expressive enough that one can develop modules of systems without regard to following any particular overall design: thus, for handling certain phenomena it may be more convenient to think in terms of a two-level system. For others, it may be easier to write cascaded rules. No matter: the two components can be combined as if one had built them both in one way or the other.

While Koskenniemi certainly did not invent finite-state approaches to morphology and phonology, he was the first to develop a system that worked fully using finite-state techniques, and he is thus to be given much credit for bringing the field of finite-state morphology to maturity, and building the way for the renaissance of finite-state approaches to language and speech that has de-

veloped over the past couple of decades.

13.3 Koskenniemi's Contributions to Indus Valley Studies

Koskenniemi's other contribution of interest here is his collaboration with the Indologist Asko Parpola in attempts to decipher the so-called Indus script (Koskenniemi and Parpola, 1980, 1982, Koskenniemi, 1981, Parpola, 1994). One of the products of that collaboration was the development of a concordance of Indus inscriptions (Koskenniemi and Parpola, 1979, 1982) that expanded on earlier work by Parpola and Koskenniemi's brother Seppo. Later on we will say a bit about that concordance, whose structure relies on the traditional assumption that the Indus symbols were part of a writing system, which we have recently challenged on a variety of statistical and non-statistical grounds (Farmer et al., 2004). Of deeper interest in the context of this paper is Koskenniemi's work on the automatic derivation of groupings among Indus symbols, which (on the linguistic assumption) he links to putative syntactic structures and to the detection in the inscriptions of possible homographs. Koskenniemi uses two main methods to distinguish sign groupings in the inscriptions. The first he attributes to S. Koskenniemi et al. 1970. The method, as Kimmo Koskenniemi describes it, involves comparing the actual counts of paired symbols with the expected counts based on the general frequencies of each sign. Symbol pairs with higher ratios are assumed to reflect underlying syntactic regularities in the system. This measure is related to pointwise mutual information (Shannon, 1948), which has been used extensively in computational linguistics for computing associations between words; for example, the measure was used in (Sproat and Shih, 1990) for the unsupervised discovery of word boundaries in Chinese texts and for parsing more generally in (Magerman and Marcus, 1990). Unfortunately, mutual information does not provide a solid foundation for syntactic analysis since high mutual information between terms is more often indicative of semantic association than syntactic constituency. While strong syntactic associations are sometimes found with closely linked terms, strong semantic associations also show up between terms that have no necessary syntactic relationship, e.g. between the English words *doctor* and *nurse*. Moreover, strong pairwise associations also show up often in non-linguistic strings, as witnessed in mathematical equations or chains of non-linguistic symbols associated with pantheons of gods (see (Farmer et al., 2004)), that have nothing to do with linguistic syntax.

Koskenniemi's other method ultimately derives from the work of Zellig Harris (1951). Starting from the left or right end of a sequence of glyphs, one counts, for each initial substring, the number of other texts that share the same beginning or end. One expects the number of possible next signs

to rise at a major syntactic boundary, since there are fewer restrictions across constituents than within constituents. Harris originally used essentially the same measure to determine the location of morph boundaries in unsegmented sequences of text. Koskenniemi argues that the two methods — the mutual-information-like method and the Harrisian method — produce similar syntactic analyses.

Koskenniemi also associates with Harris his method for detecting potential homographs. As Koskenniemi correctly notes, early writing systems were replete with homography, so it is reasonable to expect that Indus signs (based again on the assumption that they are linguistic) would also contain many homographs. The discovery of homographs is one of the trickiest aspects of decipherment. Based on what we know of the extensive homography of early scripts, we certainly cannot assume that a particular sign always has the same value; but at the same time we cannot simply assign homographs at will since such a strategy permits an unlimited number of potential decipherments of a given inscription with no obvious way to choose between them. Many of the well over 100 claimed decipherments that have been proposed in the past of the so-called Indus script have been plagued by this problem. The result is that a robust, replicable method for detecting potential homographs would be a useful tool in helping to select between potential linguistic readings of an undeciphered script. The method that Koskenniemi proposes to deal with this problem can be summarized as follows: consider symbols y and z, which occur in distinct linguistic environments, e.g. in two differing sets of preceding and following glyph environments. Now suppose one finds a glyph x that occurs in both of these environments: since x behaves in some cases like y and in other cases like z, x is a reasonable candidate for being a homograph. In other words, it is possible in this context that x is being used to represent two distinct linguistic entities. To provide an example from English, consider the words *carp* and *violin*. If one examines a corpus of English, one will likely find that the linguistic environments in which the word *carp* shows up have little in common with those that include the word *violin*. Now consider the word *bass*. If one looks again at the corpus, one will find that *bass* occurs both in environments similar to those in which *carp* appears and in environments similar to those in which we find *violin*. From this one can guess that *bass* is a potential homograph with two very different senses — in this case involving fish and musical instruments. A more sophisticated approach to automatic ambiguity detection along the lines of what Koskenniemi proposed, following Harris, was explored in (Sproat and van Santen, 1998).

The two problems that Koskenniemi addressed — the automatic detection of syntactic structures and of potential homographs — are topics that remain at the forefront of computational linguistics, as researchers search for more powerful automatic method of analyzing linguistic data. Unfortunately,

Koskenniemi's proposed methods have not had a major impact on Indus research, not due necessarily to any formal flaws in those methods, but instead, as suggested earlier, since those methods overlay the deeper, unexamined, assumption that Indus inscriptions encoded natural language. Non-linguistic sign systems often display levels of formal structure no less extreme than those seen in linguistic systems: witness the complex syntactic structures in mathematical expressions, or the recurrent sign groups that regularly show up in non-linguistic sign systems in the ancient Near East (Farmer et al., 2004). It has also long been known that non-linguistic signs display semantic "multivocality" that can be loosely pictured as the non-linguistic equivalent of homography in scripts. The upshot is that while Koskenniemi's methods may in fact identify genuine systematic relationships between symbols in Indus inscriptions, relationships of this type are not unique to writing but show up as well in a much wider class of sign systems. Since ethnographical studies suggest that the intended sense of nonlinguistic symbols are typically less "fixed" than those in written systems (cf., e.g., Barth (1987)), this finding also raises questions about the utility of the types of concordances of Indus inscriptions to which Koskenniemi has contributed, which overly standardize signs in ways that may mask important visual clues to the original sense of those signs, which may have differed widely in different Indus sites and periods as well as on diverse artifact types.

It is noteworthy that no unsupervised means has ever been proposed to distinguish linguistic from non-linguistic strings. It would be interesting to see whether the methods that Koskenniemi introduced in his studies of Indus signs might be applied to this interesting and still undeveloped area of research, grounded perhaps on systematic comparison of the specific types of regularities found in a significant cross-section of different classes of linguistic and non-linguistic sign systems. Those methods may also have possible applications in future studies of Indus symbols that are not tied to the traditional assumption, which is now being seriously challenged, that Indus inscriptions systematically encoded speech.

13.4 Summary

Koskenniemi has made many contributions to many areas in computational linguistics. This paper has reviewed what is certainly his best known contribution — two-level computational morphology, and what may well be his least-known contributions, namely his work on the Indus Valley corpus. Two-level morphology has, of course, been highly influential and is still used despite the fact that one of the main motivations for this approach (the processing power of early 1980's computers) is no longer relevant. Koskenniemi's work on the Indus Valley corpus is also interesting since, although we believe

there is compelling evidence that the Indus Valley "script" did not encode a language, he was investigating issues — the automatic discovery of structure, and the automatic discovery of senses — which are very much relevant today.

References

Barth, Frederik. 1987. *Cosmologies in the Making: A Generative Approach to Cultural Variation in Inner New Guinea*. Cambridge: Cambridge University Press.

Bird, Steven and T. Mark Ellison. 1994. One-level phonology: Autosegmental representations and rules as finite automata. *Computational Linguistics* 20(1):55–90.

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. New York: Harper and Row.

Coleman, John. 1992. *Phonological Representations — their Names, Forms and Powers*. Ph.D. thesis, University of York.

Farmer, Steve, Richard Sproat, and Michael Witzel. 2004. The collapse of the Indus-script thesis: The myth of literate Harappan civilization. *Electronic Journal of Vedic Studies* 11(2).

Harris, Zellig. 1951. *Methods in Structural Linguistics*. Chicago: University of Chicago Press.

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

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

Karttunen, Lauri, Ronald Kaplan, and Annie Zaenen. 1992. Two-level morphology with composition. In *COLING-92*, pages 141–148. COLING.

Knuth, Donald. 1973. *The Art of Computer Progamming*, vol. 3. Reading, MA: Addison-Wesley.

Koskenniemi, Kimmo. 1981. Syntactic methods in the study of the Indus script. *Studia Orientalia* 50:125–136.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production*. Ph.D. thesis, University of Helsinki, Helsinki.

Koskenniemi, Kimmo and Asko Parpola. 1979. Corpus of texts in the Indus script. Tech. rep., University of Helsinki. Department of Asian and African Studies, Research Reports 1.

Koskenniemi, Kimmo and Asko Parpola. 1980. Documentation and duplicates of the texts in the Indus script. Tech. rep., University of Helsikni. Department of Asian and African Studies, Research Reports 2.

Koskenniemi, Kimmo and Asko Parpola. 1982. Corpus of texts in the Indus script. Tech. rep., University of Helsinki. Department of Asian and African Studies, Research Reports 3.

Koskenniemi, Seppo, Asko Parpola, and Simo Parpola. 1970. A method to classify characters of unknown ancient scripts. *Linguistics* 61:65–91.

Magerman, David and Mitchell Marcus. 1990. Parsing a natural language using mutual-information statistics. In *Proceedings of Eighth Annual Meeting of AAAI*.

Mohri, Mehryar and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *Association for Computational Linguistics, 34th Annual meeting*, pages 231–238. ACL, Santa Cruz, CA.

Parpola, Asko. 1994. *Deciphering the Indus Script*. Cambridge: Cambridge University Press.

Prince, Alan and Paul Smolensky. 1993. Optimality theory. Tech. Rep. 2, Rutgers University, Piscataway, NJ.

Roark, Brian and Richard Sproat. 2006. *Computational Approaches to Syntax and Morphology*. Oxford: Oxford University Press. Forthcoming.

Shannon, Claude. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27:379–423, 623–656.

Sproat, Richard and Chilin Shih. 1990. A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese and Oriental Languages* 4:336–351.

Sproat, Richard and Jan van Santen. 1998. Automatic ambiguity detection. In *Proceedings of the International Conference on Spoken Language Processing*. Sydney.