ON THE GENERATIVE POWER OF TWO-LEVEL MORPHOLOGICAL RULES

Graeme Ritchie
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
Scotland

ABSTRACT

Koskenniemi's model of two-level morphology has been very influential in recent years, but definitions of the formalism have generally been phrased in terms of a compilation (sometimes left unspecified) into a form of finite-state transducers, or else have consisted of an informal outline of the intended interpretation of the rule-formalism itself. Analyses of the properties of the formalism have generally focussed on the transducer mechanism. It is, however, possible to give a fully formal definition of the original rule notation directly, in a way which reflects Koskenniemi's original informal characterisation and which does not depend directly on the notion of a transducer (although it must retain the essential nature of parts of the notation as being regular expressions). This re-formulation allows a proof that the ability of this formalism to characterise mappings between strings is more limited than that of arbitrary transducers.

BACKGROUND

Koskenniemi(1983a,b;1984) proposed a rule-system for describing morphological regularities in a language, depending centrally on the idea of matching two sequences of symbols - a *lexical* string (made up of the lexical forms of morphemes) and a *surface* string (the sequence of characters in the normal, inflected, form of the word). In principle, symbols could be orthographic or phonological; here we shall follow common practice within two-level morphology, and assume orthographic forms are being analysed. Koskenniemi(1983a) originally described the rules in two alternative forms - high-level rules and finite-state transducers, and conjectured that an automatic compilation procedure could be devised to transform the more readable, high-level form into the more directly implementable transducer form. His implementation was an interpreter for the transducers, which were directly written by the linguist as rules in their own right. The various linguistic analyses presented in Dalrymple et al.(1983) also follow this approach, expressing rules as transition tables for transducers. Koskenniemi(1985) refined the notation and sketched a compilation method, and Ritchie, Black et al.(1987), Karttunen et al.(1987) describe compilation techniques for two variants of the notation.

The aim of this paper is to give an alternative statement of the meaning of the original high-level rule notation, without recourse to compilation into finite-state transducers. The benefits of this are twofold:

(i) alternative implementation techniques can be considered or discussed with reference to a standard interpretation which is not tied to an existing approach to implementation;

(ii) the formal properties of the actual rule-formalism can be assessed, rather than the formal properties of another formalism (transducers) which might in principle be more powerful.

In particular, it is possible to show that the two-level morphological mechanism is more limited than the transducer model in its ability to define relationships between strings.

THE FORMALISM

TWO-LEVEL NOTATION

The original notation proposed in Koskenniemi(1983a) included some rather complex notational conventions which have not survived into later versions. The formalisation given here will deal only with the core ideas, as embodied in Koskenniemi(1985) (and other implementations such as Karttunen et al.(1987), Ritchie, Black, et al.(1987)). By way of illustration, here is a two-level morphological rule taken from Ritchie, Pulman et al. (1987):

```
e:0  <=> =:C2 ___ < +:0 V:= >
     or < C:C V:V> ___ <+:0 e:e>
     or {g:g c:c} ___ <+:0 {e:e i:i} >
     or l:0 ___ +:0
     or c:c ___ <+:0 a:0 t:t>
```

Rules are phrased in terms of symbol-pairs (written with an infix colon), where the first in the pair is a lexical symbol and the second is a surface symbol. In the above example, the pair of symbols on the left (lexical "e" and surface null) are allowed to occur only in the contexts listed on the right of the rule, where "___" indicates the position of the pair "e:0". Each context has a left part and a right part, each of these being essentially a regular expression over symbol-pairs, where angle brackets indicate sequences of pairs and braces indicate alternatives (disjunction). Certain versions of the notation may also allow the "Kleene star" symbol "*" to indicate zero or more repetitions, and the insertion of optional elements. In this example, "C", "V", "C2", and "=" represent subsets of the relevant symbol alphabets and "+" is an abstract symbol occurring in certain lexical forms.

The formalism here will not include symbolic mnemonics for sets of symbols, nor variables ranging over sets of symbols. The semantics of both these notations (which are commonly used in two-level morphology) can be stated in terms of equivalent sets of rules without such abbreviatory conventions, so all that is required is a definition of the interpretation of rules containing only actual character symbols, together with the various devices for indicating disjunction, repetition, etc. (Most of the latter could also be ignored here by a similar assumption, but the presentation is perhaps easier to follow if the resemblance to the actual notation is retained).

One of the more peripheral aspects of two-level morphology is the role of the rules in segmenting surface input strings into lexical forms (i.e. the interface between a rule interpreter and a lexicon of morphemes). It is only there that the special null symbol "0" takes on special significance (see later section). Hence most of the definitions, and the subsequent discussion of generative power, are concerned with sequences of symbol-pairs, which is equivalent to considering only pairs of strings of equal length.

BASIC DEFINITIONS

Given any two finite symbolic alphabets, A and A', a *symbol-pair* from A and A' is a pair <a, a'> where a ∈ A and a' ∈ A'. Such symbol-pairs will normally be written as "a:a'". A *symbol-pair sequence from A and A'* is simply a sequence (possibly empty) of symbol-pairs from A and A', and a *symbol-pair language over A and A'* is a set of symbol-pair sequences (i.e. a subset of (A x A')*).

Given two alphabets A and A', and a symbol-pair sequence S from A and A', a sequence $<P_1,..P_n>$ of symbol-pair sequences from A and A' is said to be a *partition* of S iff

S = $P_1P_2...P_n$ (i.e. the concatenation of the P_i)

CONTEXTS AND RULES

Given two symbol sets A and A', a *context-expression from A and A'* is a regular expression over A x A'. That is, a context-expression characterises a regular set of sequences of symbol-pairs. For example, the expression

b:b v (a:a b:b)*

characterises the set

{ε, b:b, a:a b:b, a:a b:b a:a b:b, ...}

where ε denotes the empty sequence.

Given two alphabets A and A', a *two-level morphological rule* over A and A' consists of a pair <P, C> where P is a symbol-pair from A and A', and C is a non-empty set of pairs <LC,RC> where LC and RC are context-expressions from A and A'. The reason for including a *set* of pairs of contexts, is that we must cater, in the general case, for there being a disjunction of pairs of contexts (as in the illustrative example above, where the disjuncts are separated by "or"). In the case where the set is a singleton, this reduces to the simple (non-disjunctive) case.

A context-expression ce is said to *match at the right-end* a symbol-pair sequence S iff there is a partition $<P_1, P_2>$ of S such that P_2 is an element of the set characterised by ce.

A context-expression ce is said to *match at the left-end* a symbol-pair sequence S iff there is a partition $<P_1, P_2>$ of S such that P_1 is an element of the set characterised by ce.

In a two-level morphological grammar, there are generally three sorts of rule, although one of them can be re-expressed as a combination of rules of the two more basic sorts. The first basic form of rule is the *context restriction rule* written with the operator "=>" separating the symbol-pair from the specification of the contexts. For example,

l:i => b:b ___ e:e

would mean "if there is a lexical l paired with a lexical i, then there must be a lexical and surface b on its left, and a lexical e and surface e on its right".

On the other hand, a *surface coercion rule*, written using the operator "<=" indicates that wherever the contexts (i.e. the right side of the rule) occur, *and* the lexical symbol is as given in the pair on the left side of the rule, then the surface symbol must be as given on the left side of the rule. For example:

l:i <= b:b ___ e:e

would mean that "whenever there is a lexical b and surface b on the left, a lexical e and surface e on the right, and a lexical l, then the surface symbol must be i".

The third type of rule, illustrated earlier, uses the "<=>" operator, and is defined to be equivalent to a pair of rules, one of each of the two basic types, but with the same content. Hence, no formal definition will be given of the third type of rule, on the grounds that a grammar written using the "<=>" operator is merely an abbreviation for a larger set of rules of the two basic types. We will first define the form of restriction imposed by rules normally written with the "=>" operator ("context restriction" rules).

A set R of two-level morphological rules *contextually allows* a symbol-pair sequence S iff, for every partition <P_1, a:a', P_2> of S, either there is no rule of the form <a:a', C> in R, or there is at least one rule <a:a', C> in R such that C contains a context pair <LC, RC> such that LC matches P_1 at the right end and RC matches P_2 at the left end.

The definition corresponding to a "surface coercion" rule (operator "<=") is as follows.

A two-level morphological rule R = <<a,a'>, C> *coercively allows* a symbol-pair sequence S iff for every possible partition <P_1, b:b', P_2> of S and every element <LC, RC> in C such that LC matches P_1 at the right end, and RC matches P_2 at the left end, if b = a, then b' = a'.

An alternative but equivalent variation on the last definition would be that a two-level morphological rule R = <<a,a'>, C> *coercively disallows* a symbol-pair sequence S iff there is a possible partition <P_1, b:b', P_2> of S and an element <LC, RC> in C such that LC matches P_1 at the right end, RC matches P_2 at the left end, b = a and b' ≠ a'.

TWO-LEVEL GRAMMARS

Given two alphabets A and A', a *two-level morphological grammar based on A and A'* consists of a pair <CR, SC> where CR and SC are finite sets of two-level morphological rules over A and A'. The two sets of rules are the context restriction and surface coercion rules respectively.

One minor detail which must now be considered is the question of *feasible pairs.* When set-mnemonics and variables are used within rules, these are deemed to cover not all possible symbol-pairs, but only those which are "feasible". Even when not using these abbreviatory devices, it is necessary to have some notion of feasible symbol-pair, since such pairs are allowed to occur freely even if licensed by no rule (providing no rule forbids them). Usually, pairs of the form x:x (where x is in the intersection of the two alphabets) are taken as feasible, but any pairs which appear in a rule are also deemed feasible. If we assume that the notion of a symbol-pair occurring in a regular expression is clear enough, occurrence within a rule set is straightforward— a symbol-pair a:a' is said to *occur* in a rule <b:b', C> iff either a:a' = b:b' or for at least one element <LC, RC> of C, a:a' occurs in at least one of LC and RC. Given a two-level morphological grammar G = <CR, SC>, the *set of feasible pairs in G* is the set of symbol-pairs

{a:a' |
a:a' occurs in some element of CR ∪ SC}

(In an implemented system, the user may be allowed to declare certain pairs as feasible, but at this level of abstraction we do not need to include this in our definition of a two-level morphological grammar, since such an effect could be represented by including rather vacuous context-restriction rules of the form

< a:b, { <<>,<>> } >).

Given a two-level morphological grammar G = <CR, SC>, a symbol-pair sequence S is *generated* by G iff all the following hold:
(i) all the symbol-pairs in S are feasible pairs in G;
(ii) each rule in SC coercively allows S;
(iii) the set CR of rules contextually allows S.

Notice that the two classes of rules are treated slightly differently - surface coercion rules are conjoined, forming a set of constraints all of which must be met, and context restriction rules are disjoined, giving a set of possible licensing contexts. If no rules apply to a particular symbol-pair, it is acceptable if and only if it is feasible.

THE LEXICON

The mechanisms described so far have provided a way of relating one sequence of symbols to another sequence (of the same length). There has been little or no asymmetry between the roles played by the two sequences, and no explicit indication of how these rules might achieve the practical task of segmenting a word into a set of lexical forms which appear in a given dictionary.[1] The first convention that is needed is quite simple - the string of lexical symbols is regarded as being supplied by any valid concatenation of lexical forms. That is, the set of lexical entries implicitly defines an infinite set of strings of indefinite length, formed by any concatenation of lexical forms. It is in the course of integrating the string-matching with the segmentation that the special null symbol will be needed, so we must first define the notion of two strings being the same after the removal of nulls.

Suppose we have some symbolic alphabet A. We define the function "delete" from A x A* to A* as follows, where ε denotes the empty string:

delete(a, ε) = ε
delete(a, aS) = delete(a, S)
delete(a, bS) = b delete(a,S) for any b ≠ a.

[1] The formal arguments concerning generative power concern only the mechanisms presented so far, so readers uninterested in the interface to the lexicon may skip this section.

The other minor formal definition we need is to allow us to move from equal-length sequences of symbol-pairs to pairs of equal-length symbol-sequences in the obvious way. Suppose S_1 and S_2 are two sequences of symbols, of equal length, with $S_1 = a_1...a_n$ and $S_2 = b_1...b_n$. Then the *symbol-pair sequence associated with* S_1 and S_2 is the sequence

$a_1{:}b_1....a_n{:}b_n$

We can then define a two-level morphological grammar as *licensing* a pair of strings of equal length, iff their associated symbol-pair sequence is generated by the grammar.

A *lexical segmentation system* consists of a tuple (AL, AS, 0, L, G) where AL is a finite set (the *lexical alphabet*), AS is a finite set (the *surface alphabet*), 0 is a symbol which is not an element of AL ∪ AS, L is a set (the set of *lexical forms*) of non-null elements of AL*, and G is a two-level morphographemic grammar based on AL ∪ {0} and AS ∪ {0}.

Given a lexical segmentation system (AL, AS, 0, L, G), a string S ∈ AS* *can be segmented* as $<l_1,...l_n>$ where $l_i \in$ L for all i, if there are strings $S_1 \in$ AL*, $S_2 \in$ AS* such that the following all hold:

delete(0, S_1) = $l_1 l_2....l_n$
delete(0, S_2) = S
G licenses $<S_1, S_2>$

Notice that there is no distinguished symbol indicating a morpheme boundary or word boundary. Although the writer of the two-level rules will probably find it useful to insert certain special symbols (e.g. the "+" used in the example above), these have no special significance, and rules must be written to define how they relate to other symbols. The boundaries between morphemes are implicit in the successful match between the surface form (via the two-level rules) and the concatenated sequence of lexical forms.

CROSS-LINKED LEXICONS

In Koskenniemi(1983a) (and in the papers in Dalrymple et al.(1983)) the interface to the lexicon is slightly more complicated, since the representation of morphotactic information is built into the interface, in the following way. A lexical entry (for a single morpheme) contains one or more *continuation classes* which indicate what categories of morpheme might follow it within a valid word; for example, a noun stem is marked as allowing a noun suffix as a possible continuation. The morphemes are not held in a single, uniform dictionary, but in a set of sublexicons, where each lexicon corresponds to some single morphotactic class. Hence, when the lookup process has found a particular morpheme (say, a noun stem) by matching entries in the noun-stem sublexicon, the indication that noun-suffix is a possible continuation will cause the lookup process to continue scanning in the noun suffix sublexicon as it matches the input word from left to right. This can be rephrased in a more declarative way by stating that an input string S corresponds to a sequence of lexical forms $w_1,...w_n$ if S matches $w_1w_2 ...w_n$ (the concatenation of the forms) according to the morphographemic rules, and for each i between 1 and n-1, w_{i+1} is in a continuation class of w_i. A lexical segmentation system would then have to include a function which mapped each lexical entry to its set of continuation classes. Hence the definitions given above would have to be altered to the following.

A *lexical segmentation system* consists of a tuple (AL, AS, 0, $\{L_1,..L_n\}$, f, G) where AL is a finite set (the *lexical alphabet*), AS is a finite set (the *surface alphabet*), 0 is a symbol which is not an element of AL $\cup$ AS, $\{L_i\}$ is a finite set of finite sets of non-null elements of AL* (the *sublexicons),* f is a function which associates with each pair <w, j> (where w $\in L_j$) a subset of $\{L_1,..L_n\}$ (the *continuation class* mapping) and G is a two-level morphographemic grammar based on AL $\cup$ {0} and AS $\cup$ {0}.

Given a lexical segmentation system (AL, AS, 0, L, f, G), a string S in AS* *can be segmented* as $<l_1,...l_n>$ where l_j in $L_{g(j)}$ for each j, if there are strings S_1 in AL*, S_2 in AS* such that the following all hold:

delete(0, S_1) = $l_1l_2....l_n$
delete(0, S_2) = S
G licenses $<S_1, S_2>$
$L_{g(j+1)} \in$ f(l_j, g(j)) for each j from 1 to n-1

The advantage of introducing cross-linked lexicons is that some form of morphotactic information can be inserted directly into the lexicon, and the processing of this information incorporated into the scanning of the surface string very easily. One theoretical disadvantage is that it imposes a finite-state structure on the morphotactics, which may well be undesirable. If cross-linked sublexicons are not used, some further descriptive device is needed to express morphotactic information in a usable form, but this could be completely separate from the two-level morphology system (cf. Ritchie, Pulman et al.(1987)).

LANGUAGES GENERATED

With the above definitions, it is now possible to ask what sorts of symbol-pair languages can be characterised using a two-level morphological grammar. Here we shall ignore the issue of the interface to the lexicon, and simply consider the capacity of two-level morphological grammars to characterise sets of sequences of symbol-pairs.

Lemma 1 : Let R be a set of two-level morphological rules. Let E_1 and E_2 be symbol-pair sequences such that R contextually allows E_1, and R contextually allows E_2. Then R contextually allows the concatenation E_1E_2.

Proof : If there is no symbol-pair a:a' in E_1E_2 such that there is some rule <<a:a'>, C> in R, then R contextually allows E_1E_2 for trivial reasons. Let a:a' be a symbol-pair occurring in E_1E_2 such that there is at least one rule <<a:a',C> in R. Let $<P_1$, a:a',$P_2>$ be a partition of E_1E_2. It follows from the definitions of a partition and concatenation that either P_1 is a proper initial subsequence of E_1 and $P_2 = S_2E_2$ for some sequence S_2 (i.e. this occurrence of a:a' is in E_1), or $P_1 = E_1S_1$ for some sequence S_1 and P_2 is proper final subsequence of E_2 (i.e. this occurrence of a:a' is in E_2). That is, either $<P_1$, a:a', $S_2>$ is a partition of E_1, or $<S_1$, a:a', $P_2>$ is a partition of E_2. Assume the former is true (a symmetrical argument can be followed for the latter). Since R contextually allows E_1, for the partition $<P_1$, a:a', $S_2>$ of E_1 there is at least one rule C in R which contains at least one context-pair <LC, RC> such that LC matches P_1 at the right end and RC matches S_2 at the left end. If RC matches S_2 at the left end, then RC will also match $S_2E_2 = P_2$ at the left end. Hence, for the partition $<P_1$, a:a', $P_2>$ of E_1E_2 there is at least one rule C in R which contains at least one context-pair <LC, RC> such that LC matches P_1 at the right end and RC matches P_2 at the left end. A similar argument can be given for the occurrence of a:a' being in E_2. Since this will be true for any such a:a' in E_1E_2, R contextually allows E_1E_2.

Lemma 2 : Let R = <a:a', C> be a two-level morphological rule. Let E_1, E_2, E_3 be symbol-pair sequences such that $E_1E_2E_3$ is coercively allowed by R. Then E_2 is coercively allowed by R.

Proof : If E_2 were not coercively allowed by R, it would mean that there is a partition $<S_1$, a:b, $S_2>$ of E_2 such that for some <LC, RC> in C, LC matches S_1 at the right end, RC matches S_2 at the left end, and b $\neq$ a'. If this were the case, there would be a corresponding partition $<E_1S_1$, a:b, $S_2E_3>$ of $E_1E_2E_3$, with LC matching E_1S_1 at the right end, and RC matching S_2E_3 at the left end. This would (by definition) mean that R does not coercively allow $E_1E_2E_3$, which is not the case by hypothesis.

Corollary : Let C be a set of two-level morphological rules, all of which coercively allow a symbol-pair sequence E. Then all of the rules in C coercively allow any subsequence of E.

Lemma 3 : Let G be a two-level morphological grammar <CR, SC>, and let L(G) be the set of symbol-pair sequences generated by G. Suppose that there are sequences E_1, E_2, E_3, E_4 such that $E_2 \in$ L(G), $E_3 \in$ L(G), and $E_1E_2E_3E_4 \in$ L(G). Then $E_2E_3 \in$ L(G).

Proof : (i) Since $E_1E_2E_3E_4 \in$ L(G), all the symbol-pairs in it are feasible with respect to G, hence all the symbol-pairs in E_2E_3 are feasible.
(ii) Since E_2 and E_3 are in L(G), it follows that CR contextually allows E_2 and E_3 (by definition). By Lemma 1 above, this means that CR contextually allows E_2E_3.
(iii) Since $E_1E_2E_3E_4 \in$ L(G), it follows (by definition) that all of the rules in SC coercively allow $E_1E_2E_3E_4$. Hence, by the corollary to Lemma 2 above, all of the rules in SC coercively allow E_2E_3.
This establishes the three defining conditions for $E_2E_3 \in$ L(G).

REGULAR RELATIONS

As mentioned in the introduction, two-level grammars have historically been written in two different ways— as rules as defined here, and as sets of finite-state transducers. In the latter case, each transducer deals with some linguistic phenomenon, and a sequence of symbol-pairs is generated by the grammar if *every* transducer in the grammar accepts it. That is, the symbol-pair sequence must be in the *intersection* of the languages accepted by the transducers (viewed as acceptors); in procedural terms, this is often referred to as "having the transducers executed in parallel". Hence, when working with the transducer formalism the linguist has to devise independent transducers whose intersection is the required language.

Kaplan(1988) discussed the notion of a *regular relation*, which is, roughly speaking, a symbol-pair language which can be characterised by a regular expression of symbol-pairs. Not surprisingly, a set of symbol-pair sequences is regular if and only if it can be accepted by a finite-state transducer in the obvious way. Kaplan has developed an algebraic way of manipulating regular expressions over symbol-pairs together with ordinary regular expressions over symbols, and one of his results is that the intersection of several regular relations is also a regular relation. It follows that the symbol-pair languages accepted by the two-level transducer model are exactly the regular relations.

Kaplan also formalises the re-expression of two-level morphological rules as transducers (i.e. the compilation mentioned in the introduction above) by constructing regular relations equivalent to languages generated by individual two-level morphological rules. This re-expression is one-way - from a two-level morphological rule an equivalent regular relation can be formed.

All this suggests that the "parallel transducer" model is at least as powerful as the strict two-level grammar model defined earlier. The obvious question is whether there is a difference in power; in fact, there is:

Theorem: There are regular relations (i.e. symbol-pair languages characterised by regular expressions of symbol-pairs) which cannot be generated by any two-level morphological grammar.

Proof: This follows directly from Lemma 3 above. Any language L generated by a two-level morphological grammar must have the property that if E_2, E_3, and $E_1E_2E_3E_4$ are in L, then E_2E_3 is in L. There are regular relations which do not have this property, such as the language b:b v (a:a b:b)* mentioned earlier (which contains b:b and a:a b:b but not b:b a:a b:b, even though that sequence is a subsequence of other elements of the language).

There is another, rather trivial, difference between the power of two-level morphological rules and regular relations. According to the definitions given here, the empty sequence of symbol-pairs is in every language generated by a two-level morphological grammar, since it conforms to the definition regardless of the content of the rules. The definitions could be altered to exclude the empty sequence from every language, but it is hard to see how the rule mechanism could be used to allow the empty sequence in some languages but not others.

CONCLUSIONS

We have presented an alternative formal statement of the meaning of Koskenniemi's notation for two-level morphological rules. This definition appears to be wholly faithful to the original informal explanations of the intent of two-level morphological rules, but is independent of the expression of the rules as transducers. The generative power of two-level morphological grammars, viewed as ways of characterising sets of sequences of symbol-pairs, is less than that of arbitrary transducers, despite the fact that the transducer formulation is sometimes discussed as if it were the essential definition of the two-level model.

It now remains to determine further properties of the set of two-level generatable languages. Barton et al.(1987) have shown that the recognition problem for two-level transducers (including cross-linked lexicons) is NP-complete, and a very similar demonstration can be constructed for the two-level model defined here. Closure properties (or lack thereof) of the two-level generatable languages are yet to be proven.

ACKNOWLEDGEMENTS

This work was supported by SERC/Alvey grant GR/D/83507 (IKBS 096). I would like to thank Alan Black for useful discussions of this material.

REFERENCES

Dalrymple, Mary; Doron, Edit; Goggin, John; Goodman, Beverley; and McCarthy, John (eds) 1983 Texas Linguistic Forum 22, Department of Linguistics, University of Texas at Austin, Austin, Texas.

Barton, G. Edward; Berwick, Robert C.; and Ristad, Eric Sven 1987 *Computational Complexity and Natural Language.* MIT Press, Cambridge, Mass.

Kaplan, Ronald M. 1988 Talk on finite-state transducers given at the Alvey Workshop on Parsing and Pattern Recognition, Oxford, April 1988.

Karttunen, Lauri; Koskenniemi, Kimmo; and Kaplan, Ronald M. 1987 A Compiler for Two-level Phonological Rules. Unpublished manuscript.

Koskenniemi, Kimmo 1983a Two-level Morphology: a general computational model for word-form recognition and production. Publication No.11, University of Helsinki, Finland.

Koskenniemi, Kimmo 1983b Two-level model for morphological analysis. Pp. 683-685 in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe.

Koskenniemi, Kimmo 1984 A General Computational Model for Word-Form Recognition and Production. Pp. 178-181 in *Proceedings of COLING-84* (10th International Conference on Computational Linguistics/22nd Annual Meeting of the ACL), Stanford, CA.

Koskenniemi, Kimmo 1985 Compilation of Automata from Morphological Two-Level Rules. Pp. 143-149 in *Papers from the Fifth Scandinavian Conference of Computational Linguistics* , Publication No.15, University of Helsinki, Finland.

Reape, Mike; and Thompson, Henry 1988 Parallel Intersection and Serial Composition of Finite State Transducers. Pp.535-539 in *Proceedings of COLING-88 (12th International Conference on Computational Linguistics).* Bonn.

Ritchie, Graeme D.; Black, Alan W.; Pulman, Stephen G.; and Russell Graham J. 1987 The Edinburgh/Cambridge Morphological Analyser and Dictionary System: System Description. Version 3.0. Software Paper 11, Department of Artificial Intelligence, University of Edinburgh.

Ritchie, Graeme D.; Pulman, Stephen G.; Black, Alan W.; and Russell, Graham J. 1987 A Computational Framework for Lexical Description. *Computational Linguistics 13*, (3-4):290-307.