Kimmo Koskenniemi
Instit för allmän språkvetenskap
Helsingfors universitetet
Regeringsgatan 11-13
SF-00100 Helsingfors 10
Finland

A GENERAL COMPUTATIONAL MODEL FOR WORD-FORM RECOGNITION AND PRODUCTION

1. Generative phonology as a general formalism

The formalism of generative phonology has been widely used since its introduction in the 1960's. The generative formalism is general enough to be applied to the morphology of any language, and its rules are stated in linguistically relevant terms. The morphology of a language is described by a set of rules which start from a underlying lexical representation, and transform it step by step until the surface representation is reached. So-called abstract phonology insists on invariant lexical representations for morphemes, and thus all variations among distinct surface forms must be accounted for by rules. This has led to a need for regulating the order in which the rules may be applied.

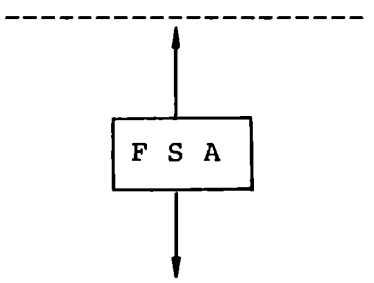
The generative formalism is conceptually unidirectional, because only the production of word-forms is guaranteed to be straight forward. As the rules are applied, they sometimes deform the context of other rules. Backwards application of rules would require either foresight or extensive trials of tentative rule applications.

The generative formalism has proven to be computationally difficult, and therefore it has found little use in morphological programs. Until recently only simulators of rules have been written and used for testing phonological descriptions or in the teaching of phonology.

A General Computational Model for Word-Form Recognition and Production
Kimmo Koskenniemi
Proceedings of NODALIDA 1983, pages 145-154

2. The computational model of Kay and Kaplan

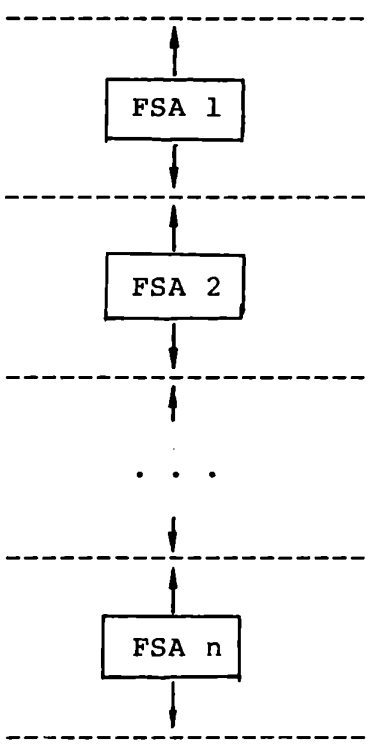
Martin Kay and Ron Kaplan from Xerox PARC noticed that each of the generative rewriting rules can be represented by a finite state automaton (or transducer) (Kay 1982). Such an automaton would compare two successive levels of the generative framework: the level immediately before application of the rule, and the level after application of the rule:



the string before application of the rule

the string after application of the rule

The whole morphological grammar would then be a cascade of such levels and automata:



lexical representation

after 1st rule

after 2nd rule

after (n-1)st rule

surface representation

A cascade of automata is not operational as such, but Kay and Kaplan noted that the automata could be merged into a single, larger automaton. Merging is possible by using the techniques of automata theory. The large automaton would be functionally identical to the cascade, although single rules could no more be identified within it. The merged automaton would be both operational, efficient and bidirectional. Given a lexical representation, it would produce the surface form, and, vice versa, given a surface form it would guide lexical search and locate the appropriate endings in the lexicon.

In principle, the approach seems ideal. But there is one vital problem: the size of the merged automaton. Descriptions of languages with complex morphology, such as Finnish, seem to result in very large merged automata. Although there are no conclusive numerical estimates yet it seems probable that the size may grow prohibitively large.

3. The two-level approach

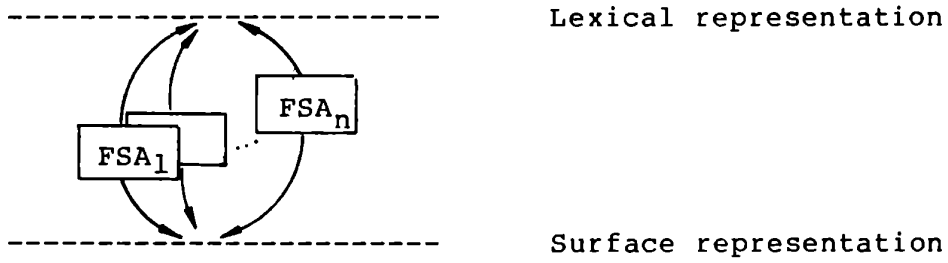
My approach is computationally very close to that of Kay and Kaplan, but it is based on a different morphological theory. It avoids the problems of their model. Instead of abstract phonology, I follow the lines of concrete or natural morphology (e.g. Linell, Jackendoff, Zager, Dressler, Wurzel).

The two-level model rejects abstract lexical representations, i.e. there need not always be a single invariant underlying representation. Some variations are considered suppletion-like and are not described with rules. The role of rules is restricted to one segment variations, which are fairly natural. Alternations which affect more than one segment, or where the alternating segments are unrelated, are considered suppletion-like and handled by the lexicon system.

The rules are completely parallel in the two-level model. The principle leads to a different theory of morphology. The new theory is incompatible with abstract phonology, but in agreement with (at least certain branches of) concrete/natural morphology.

4. Two-level rules and automata

There are only two representations in the two-level model: the lexical representation and the surface representation. No intermediate stages "exist", even in principle. The rules correspond to automata, as in the Kay and Kaplan model, but they operate in parallel instead of being cascaded:



The rule-automata compare the two representations directly, and a configuration must be accepted by each of them in order to be valid.

The two-level model (and the program) can operate in both directions: the same description can be utilized as such for producing surface word-forms from lexical representations, and for analyzing surface forms by finding the appropriate lexical entries and endings. In this sense the two-level model is bidirectional just as the Kay and Kaplan model.

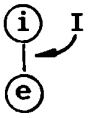
In addition to processual bidirectionality, the two-level model achieves a certain kind of conceptual nondirectionality. This means that the descriptions are not built on the concept of producing surface forms from underlying lexical representations. Instead, the rules describe the morphology and morphophonology in terms of correspondences. Thus, the two-level rules do not produce anything, they only relate two levels of representation to each other.

We take an example from Finnish morphology. The noun **lasi** 'glass' represents the productive and most common type of nouns ending in **i**. The lexical representation of the partitive plural form consists of the stem **lasi**, the plural morpheme **I**, and the partitive ending **A**. In the two-level framework we write the lexical representation **lasiIA** above the surface form **laseja**:

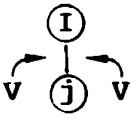
Lexical representation:	l a s i I A
Surface representation:	l a s e j a

This configuration exhibits three morphophonological variations:

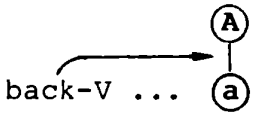
(1) Stem final **i** is realized as **e** in front of typical plural forms, i.e. when **I** follows on the lexical level, schematically:



(2) The plural **I** itself is realized as **j** if it occurs between vowels on the surface, schematically:



(3) The partitive ending, like other endings, agrees with the stem with respect to vowel harmony. An archiphoneme **A** is used instead of two distinct partitive endings. It is realized as **ä** or **a** according to the harmonic value of the stem, schematically:

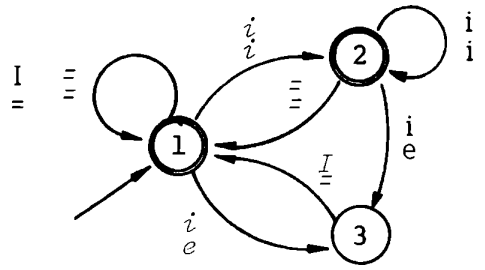


As we can see, the two-level rules may refer both to the lexical and to the surface representations.

In order to see how finite state automata can be used for describing such morphophonological alternations, we construct the three automata which perform the checking needed for the three alternations mentioned above. Instead of single characters, the automata accept character pairs. Each of the three automata must accept the following sequence of pairs:

l	a	s	i	I	A
l	a	s	e	j	a

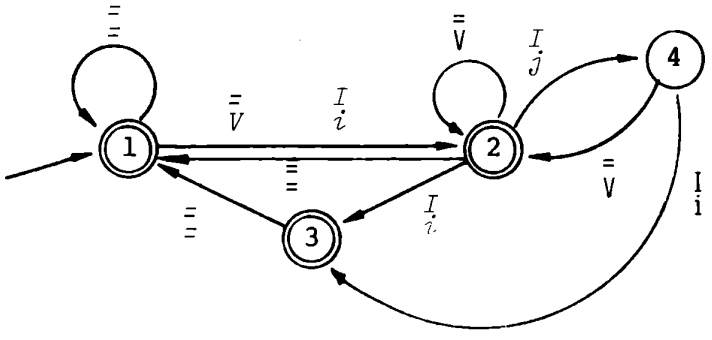
The task of the first rule-automaton is to permit the pair $\frac{i}{e}$ if and only if the plural I follows. The following automaton with three states (1, 2, 3) performs this:



State 1 is the initial state of the automaton. If the automaton receives pairs other than $\frac{i}{e}$ or $\frac{i}{i}$ it will remain in state 1 (the symbol $\frac{=}{=}$ denotes "any other pair"). Receiving a pair $\frac{i}{e}$ causes a transition to state 3. States 1 and 2 are final states (denoted by double circles), i.e. if the automaton is in one of them at the end of the input, the automaton accepts the input. State 3 is, however, a nonfinal state, and the automaton should leave it before the input ends (or else the input is rejected). If the next character pair has plural I as its lexical character (which is denoted by $\frac{I}{=}$), the automaton returns to state 1. Any other pair will cause the input to be rejected because there is no appropriate transition arc. This part of the automaton accomplishes the "only if" part of the correspondence: the pair $\frac{i}{e}$ is allowed only if it is followed by the plural I.

The remaining state 2 is needed for the converse statement. If a lexical i is followed by plural I, we must have the correspondence $\frac{i}{e}$ and nothing else on the surface. Thus, if we encounter a correspondence of lexical i other than $\frac{i}{e}$ (these are denoted by $\frac{i}{=}$) it must not be followed by the plural I. Anything else ($\frac{=}{=}$) will return the automaton to state 1.

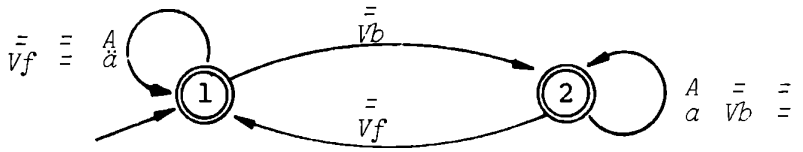
The automaton for plural I between vowels consists of four states:



The automaton remains in the initial state 1 until it enounters a surface vowel. It advances to state 2 after a surface vowel ($\bar{\bar{V}}$), but returns to state 1 if something else (i.e. a consonant) follows. The correspondence $\frac{I}{j}$ is allowed only in state 2 (the only transition arc labelled with $\frac{I}{j}$ starts from 2). Receiving this pair leads to state 4, which is nonfinal because the right context must also be satisfied. The only escape from state 4 is via a surface vowel, anything else terminates the execution because of a missing transition arc. This part of the automaton (1, 2, 4) checks that $\frac{I}{j}$ occurs only in the correct context.

The plural I is usually realized as i on the surface, but between vowels this possibility is excluded. This is accomplished by state 3. The default correspondence $\frac{I}{i}$ may be preceded or followed by surface vowels, but not at the same time. Thus, there is an appropriate transition from 1 to 1, but from state 2 (where a vowel has been detected at the left) we go to state 3. There anything else but a surface vowel will return us to state 1. The absence of an arc for $\bar{\bar{V}}$ excludes the default occurrence between vowels.

The automaton for vowel harmony is a simple one. It has only two states, both of which are final states:



The automaton starts from state 1 and remains there if the word-form is front harmonic. Detection of a back harmonic vowel (Vb = **a, o, u**) will cause a transition to state 2. In the front harmonic state 1 only $\frac{A}{ä}$ is allowed, and in the back harmonic state 2 only $\frac{A}{a}$.

As it stands now, the two-level program accepts the rules as tabular automata, e.g. the first rule automaton is coded as:

```
"i - e in front of plural I"  3  4
                 i   i   I   =
                 i   e   =   =
           1:    2   3   1   1
           2:    2   3   0   1
           3.    0   0   1   0
```

This entry format is, in fact, more practical than the state transition diagrams. The tabular representation remains more readable even when there are half a dozen states or more. It has also proven to be quite useful even for those who are linguists rather than computer professionals, and it has been applied in descriptions for several languages (Karttunen & Wittenburg 1983, Alam 1983, Khan 1983, Lun 1983).

There is, however, an obvious need for using a more rule-like formalism instead of the automata. There is a notation for two-level rules (Koskenniemi 1983), according to which the above automaton could be written as:

$$\underset{e}{i} \quad <=> \quad -- \quad \underset{=}{I}$$

This formalism is useful in schetching two-level descriptions with pencil and paper. A compiler accepting rules in such a formalism and automatically transforming them into finite state automata has been planned.

5. Two-level lexicon system

Single two-level rules are at least as powerful as single rules of generative phonology. The rule component of the two-level model as a whole (at least in practical descriptions) appears to be less powerful, because of the lack of extrinsic rule ordering. Within the two-level model, certain types of variations are described in the lexicon rather than by using rules.

Variations affecting longer sequences of phonemes or where the relation between the alternatives is phonologically otherwise nonnatural, are described by giving distinct lexical representations. Generalizations are not lost since insofar as the variation pertains to many lexemes, the alternatives are given as a minilexicon referred to by all entries possessing the same alternation.

The alternation in words of the following types are described using the minilexicon method:

hevonen	-	hevosen		'horse'
vapaus	-	vapautena	- vapauksia	'freedom'

The lexical entries of such words gives only the nonvarying part of the stem and refers to a common alternation pattern nen/S or s-t-ks/S:

```
hevo      nen/S     "Horse S";
vapau     s-t-ks/S  "Freedom S";
```

The minilexicons for the alternation patterns list the alternative lexical representations and associate them with the appropriate sets of endings:

```
LEXICON nen/S        nen   S0    "";
                     sE    S123  ""
LEXICON s-t-ks/S     s     S0    "";
                     TE    S13   "";
                     ksE   S2    ""
```

6. Current status of the two-level program

The two-level program has been implemented first in PASCAL language and can be run at least on the Burroughs B7800, DEC-20, and large IBM systems. The program is fully operational and reasonably fast (about 0.1 CPU seconds per word although hardly any effort has been spent to optimize the execution speed). Lauri Karttunen and his students at the University of Texas have implemented the model in INTERLISP (Karttunen 1983, Gajek & al. 1983, Khan & al. 1983). The execution speed of their version is comparable to that of the PASCAL version. Nothing would prevent the use of the PASCAL version on micro-computeres with e.g. 128 kB memory.

The model has been tested by making a comprehensive description of Finnish morphology covering all types of nominal and verbal inflection including compounding. Karttunen and his students have made two-level descriptions of Japanese, Ruma-

nian, English and French. At the University of Helsinki, two descriptions are reaching completion: one of Swedish and one of Old Church Slavonic.

The two-level model could be part of any natural language processing system. Especially the ability both to analyze and to generate is useful. Systems dealing with many languages, such as machine translation systems, could benefit from the uniform language-independent formalism.

The accuracy of information retrieval systems can be enhanced by using the two-level model for discarding hits which are not true inflected forms of the search key. The algorithm could be also used for detecting spelling errors.

References

Alam, Y., 1983. A Two-Level Morphological Analysis of Japanese. TLF 22, pp. 229-253.

Gajek, O., H. Beck, D. Elder, and G. Whittemore, 1983. KIMMO: LISP Implementation. TLF 22, pp. 187-202.

Karttunen, L., 1983. KIMMO: A General Morphological Processor. TLF 22, pp. 165-186.

Karttunen, L., and K. Wittenburg, 1983. A Two-Level Morphological Description of English. TLF 22, pp. 217-228.

Kay, M., 1982. When meta-rules are not meta-rules. In: Sparck-Jones, K. and Y. Wilks, 1982. Automatic natural language Parsing. University of Essex, Cognitive Studies Centre. (CSCM-10.)

Khan, R., 1983. A Two-Level Morphological Analysis of Rumanian. TLF 22, pp. 253-270.

Khan, R., J. Liu, T. Ito, and K. Shuldberg, 1983. KIMMO User's Manual. TLF 22, pp. 203-216.

Koskenniemi, K., 1983a. Two-level Model for Morphological Analysis. Proceedings of IJCAI-83, pp. 683-685.

---, 1983b. Two-level Morphology: A General Computational Model for Word-Form Recognition and Production. University of Helsinki, Dept. of General Linguistics, Publications, 11.

Lun, S., 1983. A Two-Level Analysis of French. TLF 22, pp. 271-278.

TLF: Texas Linguistic Forum. Department of Linguistics, University of Texas, Austin, TX 78712.