A GENERAL COMPUTATIONAL MODEL FOR WORD-FORM RECOGNITION AND PRODUCTION

Kimmo Koskenniemi
Department of General Linguistics
Univeristy of Helsinki
Hallituskatu 11-13, Helsinki 10, Finland

ABSTRACT

A language independent model for recognition and production of word forms is presented. This "two-level model" is based on a new way of describing morphological alternations. All rules describing the morphophonological variations are parallel and relatively independent of each other. Individual rules are implemented as finite state automata, as in an earlier model due to Martin Kay and Ron Kaplan. The two-level model has been implemented as an operational computer programs in several places. A number of operational two-level descriptions have been written or are in progress (Finnish, English, Japanese, Rumanian, French, Swedish, Old Church Slavonic, Greek, Lappish, Arabic, Icelandic). The model is bidirectional and it is capable of both analyzing and synthesizing word-forms.

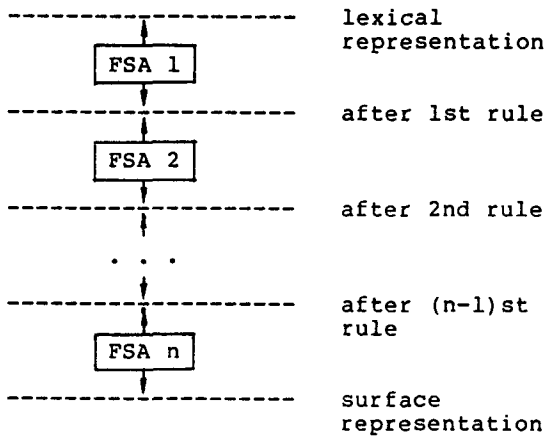
1. Generative phonology

The formalism of generative phonology has been widely used since its introduction in the 1960's. The morphology of any language may be described with the formalism by constructing a set of rewriting rules. The rules start from an underlying lexical representation, and transform it step by step until the surface representation is reached.

The generative formalism is unidirectional and it has proven to be computationally difficult, and therefore it has found little use in practical morphological programs.

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



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 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 close to that of Kay and Kaplan, but it is based on a different morphological theory. In-

---

The work described in this paper is a part of the project 593 sponsored by the Academy of Finland.

stead of abstract phonology, I follow the lines of concrete or natural morphology (e.g. Linell, Jackendoff, Zager, Dressler, Wurzel). Using this alternative orientation I arrive at a theory, where there is no need for merging the automata in order to reach an operational system.

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.

4. Two-level rules

There are only two representations in the two-level model: the lexical representation and the surface representation. No intermediate stages "exist", even in principle. To demonstrate this, 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**:

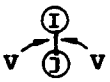
Lexical representation:	l	a	s	i	I	A
Surface representation:	l	a	s	e	j	a

This configuration exhibits three morphophonological variations:

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

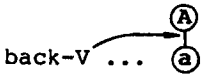
(1)

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



(2)

c) 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:



(3)

The task of the two-level rules is to specify how lexical and surface representations may correspond to each other. For each lexical segment one must define the various possible surface realizations. The rule component should state the necessary and sufficient conditions for each alternative. A rule formalism has been designed for expressing such statements.

A typical two-level rule states that a lexical segment may be realized in a certain way if and only if a context condition is met. The alternation (1) in the above example can be expressed as the following two-level rule:

$$\begin{matrix} i \\ e \end{matrix} \quad <=> \quad \text{---} \quad \begin{matrix} I \\ = \end{matrix} \qquad (1')$$

This rule states that a lexical i may be realized as an e only if it is followed by a plural I, and if we have a lexical i in such an environment, it must be realized as e (and as nothing else). Both statements are needed: the former to exlude i-e correspondences occurring elsewhere, and the latter to prevent the default i-i correspondence in this context.

Rule (1') referred to a lexical segment I, and it did not matter what was the surface character corresponding to it (thus the pair I-=). The following rule governs the realization of I:

$$\begin{matrix} I \\ j \end{matrix} \quad <=> \quad \begin{matrix} = \\ V \end{matrix} \quad \text{---} \quad \begin{matrix} = \\ V \end{matrix} \qquad (2')$$

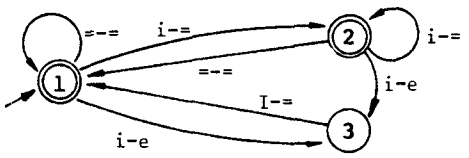
This rule requires that the plural I must be between vowels on the surface. Because certain stem final vowels are realized as zero in front of plural I, the generative phonology orders the rule for plural I to be applied after the rules for stem final vowels. In the two-level framework there is no such ordering. The rules only state a static correspondence relation, and they are nondirectional and parallel.

5. Rules as automata

In the following we construct an automaton which performs the checking needed for the i-e alternation discussed above. Instead of single characters, the automaton accepts character pairs. This automaton (and the automata for other rules) must accept the following sequence of pairs:

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

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

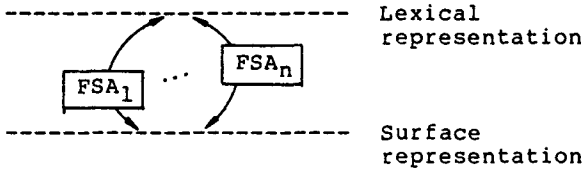


(1")

State 1 is the initial state of the automaton. If the automaton receives pairs without lexical i it will remain in state 1 (the symbol =-= denotes "any other pair"). Receiving a pair 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 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 i-e is allowed only if it is followed by the plural I.

The state 2 is needed for the "if" part. If a lexical i is followed by plural I, we must have the correspondence i-e. Thus, if we encounter a correspondence of lexical i other than i-e (i-=) it must not be followed by the plural I. Anything else (=-=) will return the automaton to state 1.

Each rule of a two-level description model corresponds to a finite state automaton as in the model of Kay and Kaplan. In the two-level model the rules or the automata operate, however, in parallel instead of being cascaded:



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

The two-level model (and the program) operates in both directions: the same description is utilized as such for producing surface word-forms from lexical representations, and for analyzing surface forms.

As it stands now, two-level programs read the rules as tabular automata, e.g. the automaton (1") is coded as:

"i - e in front of plural I" 3 4

	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 feasible even for those who are linguists rather than computer professionals.

Although it is feasible to write morphological descriptions directly as automata, this is far from ideal. The two-level rule formalism is a much more readable way of documenting two-level descriptions, even if hand compiled automata are used in the actual implementation. A compiler which would accept rules directly in some two-level rule formalism would be of great value. The compiler could automatically transform the rules into finite state automata, and thus facilitate the creation of new descriptions and further development of existing ones.

5. Two-level lexicon system

Single two-level rules are at least as powerful as single rules of generative phonology. The two-level rule component as a whole (at least in practical descriptions) appears to be less powerful, because of the lack of extrinsic rule ordering.

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:

hevo<u>nen</u>	-	hevo<u>s</u>en	'horse'
vapau<u>s</u>	-	vapau<u>t</u>ena	
	-	vapau<u>ks</u>ia	'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 pat-

terns 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

The two-level program has been implemented first in PASCAL language and is running at least on the Burroughs B7800, DEC-20, and large IBM systems. The program is fully operational and reasonably fast (about 0.05 CPU seconds per word although hardly any effort has been spent to optimize the execution speed). It could be used run on 128 kB micro-computeres as well. 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. The two-level model has also been rewritten in Zetalisp (Ken Church at Bell) and in NIL (Hank Bromley in Helsinki and Umeå).

The model has been tested by writing a comprehensive description of Finnish morphology covering all types of nominal and verbal inflection including compounding (Koskenniemi, 1983a,b). Karttunen and his students have made two-level descriptions of Japanese, Rumanian, English and French (see articles in TLF 22). At the University of Helsinki, two comprehensive descriptions have been completed: one of Swedish by Olli Blåberg (1984) and one of Old Church Slavonic by Jouko Lindstedt (forthcoming). Further work is in progress in Helsinki for making descriptions for Arabic (Jaakko Hämeen-Anttila) and for Modern Greek (Martti Nyman). The system is also used the University of Oulu, where a description for Lappish is in progress (Pekka Sammallahti), in Uppsala, where a more comprehensive French description is in progress (Anette Östling), and in Gothenburg.

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.

ACKNOWLEDGEMENTS

My sincere thanks are due to my instructor, professor Fred Karlsson, and to Martin Kay, Ron Kaplan and Lauri Karttunen for fruitful ideas and for acquainting me with their research.

REFERENCES

Alam, Y., 1983. A Two-Level Morphological Analysis of Japanese. In TLF 22.

Blåberg, O., 1984. Svensk böjningsmorfologi: en tillämpning av tvånivåmodellen. Unpublished seminar paper. Department of General Linguistics, University of Helsinki.

Gajek, O., H. Beck, D. Elder, and G. Whittemore, 1983. KIMMO: LISP Implementation. In TLF 22.

Karlsson, F. & Koskenniemi, K., forthcoming. A process model of morphology and lexicon. Folia Linguistica.

Karttunen, L., 1983. KIMMO: A General Morphological Processor. In TLF 22.

Karttunen, L. & Root, R. & Uszkoreit, H., 1981. TEXFIN: Morphological analysis of Finnish by computer. A paper read at 71st Meeting of the SASS, Albuquerque, New Mexico.

Karttunen, L. & Wittenburg, K., 1983. A Two-Level Morphological Description of English. In TLF 22.

Kay, M., 1982. When meta-rules are not meta-rules. In Sparck-Jones & Wilks (eds.) Automatic natural language processing. University of Essex, Cognitive Studies Centre. (CSM-10.)

Khan, R., 1983. A Two-Level Morphological Analysis of Rumanian. In TLF 22.

Khan, R. & Liu, J. & Ito, T. & Shuldberg, K., 1983. KIMMO User's Manual. In TLF 22.

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, No. 11.

Lindstedt, J., forthcoming. A two-level description of Old Church Slavonic morphology. Scando-Slavica.

Lun, S., 1983. A Two-Level Analysis of French. In TLF 22.

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