Gouvernement du Canada
Ministère des Communications

Government of Canada
Department of Communications

Le Centre canadien de recherche sur l'informatisation du travail
Canadian Workplace Automation Research Centre

# Finite-state Morphology: Overview of Existing Models and Applications in Continuous-text Environments.

Michel Simard



Canada

2

# Finite-state Morphology: Overview of Existing Models and Applications in Continuous-text Environments.

Michel Simard

Assisted Translation Group

Canadian Workplace Automation
Research Centre

Communications Canada

Laval



may 1990

cat. N° Co 28-1/51-1990 E
ISBN 0-662-17911-0

The author is solely responsible for the opinions expressed in this report.

* Ce rapport est aussi disponible en français.

# Finite-state Morphology: Overview of Existing Models and Applications in Continuous-text Environments.

Michel Simard[1]

Centre canadien de recherche sur
l'informatisation du travail
Communications Canada
1575, boul. Chomedey
Laval (Québec)
CANADA H7V 2X2

## Abstract

Finite-state morphological models are formalisms for describing the set of valid word-forms of a natural language. Being well-suited for computer implementations, they have typically been used for creating systems that efficiently recognize and generate isolated word-forms. In this paper, we give a presentation of Koskenniemi's *two-level* morphological model, followed by a comparison with alternative approaches. Integrating such models to natural-language processing systems that deal with NL *sentences* typically implies modifying the morphological component so that it works in a *continuous-text* environment. We discuss how this transition from isolated-words to continuous-text may be done, and show that, in addition to the orthographical phenomena normally described by morphological models, the resulting system displays interesting properties for describing "inter-word" phenomena such as *elisions* and *contractions*.

Topics: two-level morphology; finite-state morphology; computational linguistics.

## Introduction

The interest in computational models of the morphology of natural languages is quite recent. It may be traced to the appearance of Kimmo Koskenniemi's 'two-level' model ([Kos83]), a linguistic model for morphological analysis and synthesis that is suitable for a computer implementation. It consists in a formalism for describing how words of a language are constructed from a set of morphemes. This description may then be used to generate a program that analyzes or synthesizes words of that language. It has been implemented in Pascal ([Kos83]), in Lisp ([Kar83],[DKK87]) and in Prolog ([Boi88b]), among other languages. A number of computational systems have since emerged which, given the 'surface form' (physical appearance) of a word, return a description of that word

1. Email : m_simard@ccrit.doc.cdn

derived from its decomposition into morphemes, and vice-versa. These systems are usually referred to as "finite-state models" because they use finite-state devices as parsers. Such formalisms are well suited for designing computer applications requiring lexical descriptions of highly inflected languages, such as Finnish or Turkish, as well as for 'simpler' languages such as English or French

Many of these finite-state models were originally designed to handle isolated words. When examining the possibility of integrating such models to larger natural-language processing environments (e.g. an automatic translation system, or a natural-language database interface), the following question naturally arises: what happens when finite-state morphological models are modified to deal with *continuous-text*, as is the case if they are to be used within a system that works with natural language *sentences*? Interestingly, it seems that the resulting systems are capable of handling things that most syntactic models are not very good at, and that morphological models were not primarily designed for. There are a number of phenomena that seem to be conditioned by the relative position of a word within a sentence, rather than by its syntactic role. Examples of this are elisions and contractions, liaisons in speech, or simply the appearance of 'spaces' in written text. While these phenomena are usually difficult to describe syntactically, they fall within the field of competence of one of the components of the two-level model — the one that deals with *phonology* or *orthography*.

In the first part of this paper, we examine Koskenniemi's two-level model in detail: in section 1, we present some basic concepts, then discuss the formalism in section 2, and finally consider its implementation in section 3. In section 4, we review a number of other approaches that have emerged since Koskenniemi's. The second part of the paper deals with the continuous-text question: in section 5, we give a brief sketch of how Koskenniemi's two-level rules may be used to describe *inter-word phenomena.*

## 1 Some Background

In general, a lexicon may be seen as defining the relation between the *surface forms* of the words of a language (their physical appearance in written text or speech) and their *lexical descriptions* (or *definitions*). In a computational perspective, if both sets of possible surface forms and of possible lexical descriptions are finite, then one way of implementing this relation is to create an exhaustive list of *surface-form/lexical-description* pairs that satisfy the relation, and to use some searching algorithm. If one or both sets turn out to be infinite, or simply of relatively large size, this solution becomes unacceptable for obvious reasons.

The key idea behind Koskenniemi's system is that this 'lexical relation' may be decomposed and described entirely through the descriptions of the *morphology* and *phonology* of the language, and that from these, efficient recognition/generation programs may be obtained. The point here is that in the general case, the description of these individual relations is considerably more compact than the exhaustive list suggested above, and, needless to say, finite.

Informally, morphemes are the objects words are constructed from: it is generally agreed upon that all words of a language are made up of smaller pieces, put together according to rules that define the *morphology* of the language. For example, a word such as *transported* can be seen as formed of three morphemes: prefix *trans-*, root *port*, and suffix *-ed*. A distinction is usually drawn between *derivational* and *inflectional* morphologies: 'derivation' refers to the morphological processes by which new words are obtained from older ones (e.g. *transport* from *port*), and 'inflections' refer to the various shapes that a same and unique word may take (e.g. *transport*, *transported*, *transporting*, etc.)[2].

This is further complicated by the fact that the same morpheme may 'realize' in several ways (appear under various shapes on the surface) depending on its environment, i.e. on the shape of the morphemes next to which it appears. These variations are controlled by the *phonology* of the language. In general, 'phonology' refers to these phenomena, as observed in spoken language. While it is generally accepted that phonological phenomena in speech obey fairly strict principles, their counterpart in written text (sometimes called 'orthography' in the computational linguistics literature) is seldom studied by linguists because in most languages, spelling conventions seem rather arbitrary. In spite of this, in order to simplify all that follows, we will focus our attention on orthography rather than phonology. As well, we will try as much as possible to stick to English in the examples.

## 2 The Formalism

Koskenniemi's idea is to "split the work in two", i.e. to introduce an intermediate representation, called the *lexical form*, between the surface form and the lexical description. This representation is actually the result of a concatenation of objects representing morphemes of the language. When verifying if some surface-form/lexical-description pair is in the 'lexical relation', we check that there exists a lexical form that satisfies a first relation with the surface form (the orthographical relation) and a second relation with the lexical description (the morphological relation)

The first underlying hypothesis here is that the lexical description of a word may be easily computed as a function of the descriptions of its composing morphemes[3]. To obtain these, it must be possible to decompose a word into morphemes: the second hypothesis is that this decomposition is easy to do if we have access to a representation of the word displaying it as it would appear if the orthographical process did not occur, i.e. as if orthographical rules did not affect the surface realization of this word: this is what Koskenniemi's intermediate representation corresponds to. The third hypothesis is that these orthographical rules are easy to apply and *un*-apply, so that the correspondence between the lexical and surface form is also easy to compute. The morphological relation is defined by means of a morphological lexicon, i.e. a lexicon of all morphemes of the language, along with morphological rules (which morphemes may combine with one another, and in what way), and the

2. Actually, things are not all that simple, mainly because it is not clear what exactly is meant by 'two different words' and 'the same word under two different forms', but these problems need not concern us here.

3. When using the expression "easy to compute", we refer to the intuitive notion of 'practical efficiency' rather than to that of 'tractability': while the general problem of parsing the languages defined by Koskenniemi's model has proven to be NP-complete (see [Bar86]), in most real-life cases, efficient parsers may be constructed.

orthographical relation with a set of *two-level* orthographical rules. These two components are described in more details below.

## 2.1 The Lexicon

In the two-level model, the lexicon formalism is used to define the morphemes and the morphological rules of the language. It appears as a list of entries, each of which corresponds to a morpheme, and is partitioned into *sublexicons* on the basis of the a set of principles called the *morphotactics* of the language (this partitioning of morphemes is akin to the *categorization* of words in the syntax). To each morpheme is associated a *lexical form* (its appearance in the intermediate lexical representation), a *continuation class*, and some attached information.

The lexical form of a morpheme is somehow meant to denote its *underlying canonical form*, i.e. the unique representation from which all its surface realizations are derived, according to the orthographical rules of the language. It is up to whoever is producing a lexicon to decide what these lexical forms should actually look like, but in most cases, we can expect them to differ only in minor ways from their surface counterparts: a lexical form should be something like a *generalization* of all the surface realizations of the morpheme to which it is attached. Therefore, the *lexical alphabet* of a language, i.e. the alphabet in which these forms are written, although it may be anything, will probably contain all characters of the *surface alphabet*, plus maybe a small set of additional characters (diacritics, etc.). For example, the lexical form of the English verb *to carry* could be something like *carrÿ*, where character *ÿ* is meant to denote a generalization of surface realizations *y* (as in *carry*) and *i* (as in *carried*). Yet another possibility is simply to use one of the surface realizations: in this case, *carry* would also be an acceptable lexical form for the verb *to carry*, insofar as both *y* and *i* were considered as possible surface realizations of lexical character *y*.

Each *continuation class* refers to the set of sublexicons that contain morphemes which may concatenate to the right of the current one: they are the means by which the *morphological rules* of a language are defined in this model. Some sublexicons are labeled as *root*: those contain morphemes that may 'begin' words of the language. There is also a special *final* continuation class, which refers to an empty set of sublexicons, and which is given to morphemes that may 'end' a word. Lexical forms of words of the language may then be obtained by concatenating morphemes of the lexicon in ways prescribed by the continuation classes, starting with a 'root' morpheme, and ending with a 'final' continuation class morpheme.

Such a lexicon for a (very) small subset of English appears in figure 1. Boxes denote sublexicons, ellipses denote continuation classes, morphemes are in italics (actually, the lexical forms of these morphemes), and *Vroot* and *Nroot* are the root sublexicons. Morphemes that do not point to a specific continuation class are considered to be 'final'. From this lexicon could be obtained such 'words' as

*mice*  *eat0*  *cry+ed*  *walk+er+s+'s*

Notice that in practice, the word 'morpheme' may be taken in a very loose sense when building a lexicon. For example, if it turns out to be convenient in a given context to

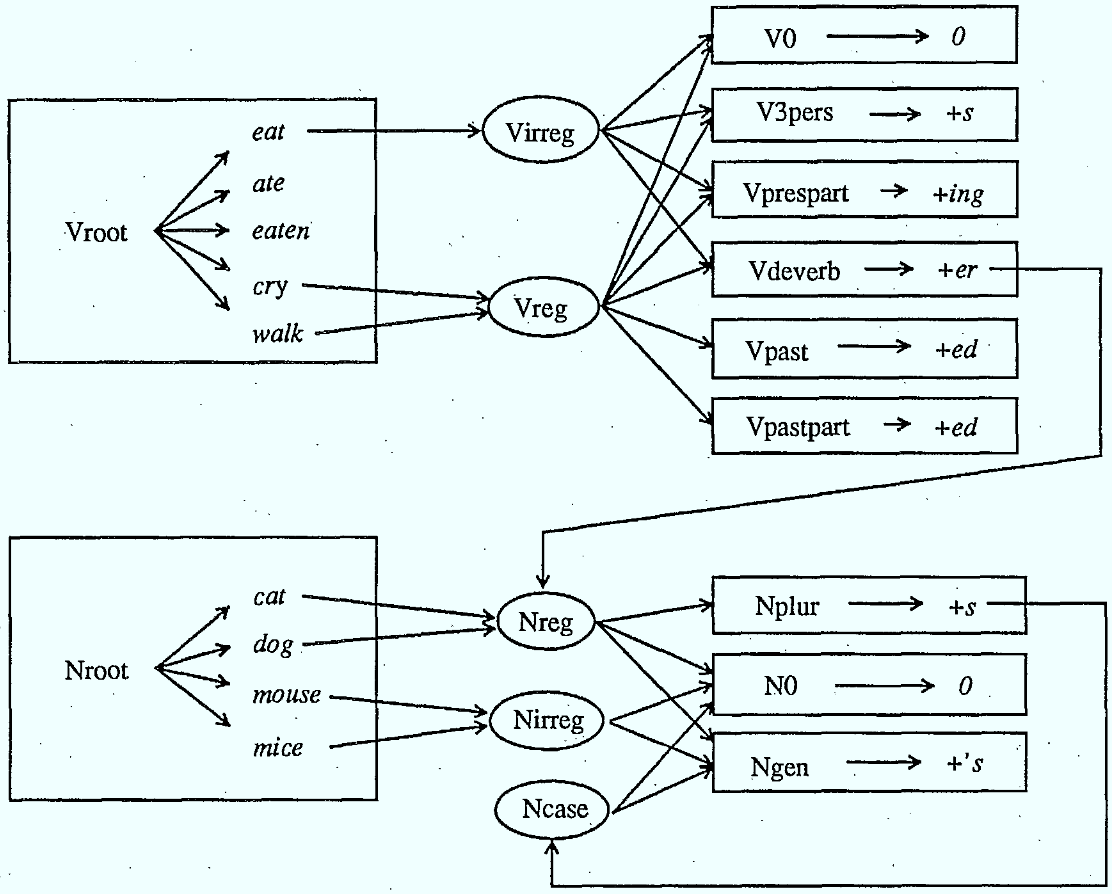


**figure 1** Sample English lexicon.

consider the expression *transporter room* as a 'morpheme', then so be it. From a purely linguistic point of view, this may be a bit shocking, but as it happens, it is common practice.

Here, the information attached to the morphemes has been omitted for clarity, but in the original model, it appears as a string of text. The lexical description of a word is the result of the concatenation of the individual descriptions of its morphemes in their order of appearance. For example, if morpheme *cry* has attached information "*verb 'to cry'*" and suffix *+ed* has "*past-tense*", then lexical form *cry+ed* has lexical description "*verb 'to cry' past-tense*". This rather primitive way of 'collecting' lexical information is a theoretical expedient whose sole purpose is to simplify the demonstration of the correctness of Koskenniemi's analyses. In later implementations of the model, the format of lexical descriptions as well as the manner in which the description of a morpheme affects that of the words it appears in are left for the user to define.

## 2.2 Two-level Rules

Two-level rules define the relations that exist between the lexical and surface forms by specifying *character-to-character* correspondences between the two representations. In this way, they define the orthographical rules of the language. They have the following format:

<*cp*> <*operator*> <*lc*> — <*rc*>

where <*operator*> is one of $\Rightarrow$, $\Leftarrow$ or $\Leftrightarrow$, <*cp*> is a *concrete pair set* (or *CPS*), and <*lc*> and <*rc*> are *regular pair expressions* (or *RPEs*), i.e. regular expressions over an alphabet of CPSs. These CPSs are pairs *(x,y)*, where *x* and *y* are elements of the lexical and surface alphabets respectively, and that describe possible realizations of lexical characters on the surface. Thus *(x,y)* may be read as "lexical character *x*, and its surface realization *y*".

Some characters have a special meaning within CPSs: the = stands for 'anything' as in *(=,s)* , to be read "any lexical character and its surface realization *s*". Conversely *(s,=)* stands for "lexical character *s* and whatever its surface realization is". Character $\varepsilon$ stands for the 'null' character: $(s,\varepsilon)$ reads "lexical character *s*, which realizes as the null string (nothing) on the surface". We also sometimes make use of variables: capital letters will normally be used for those, as in $(V,x)$, $V \in \{ a,e,i,o,u \}$. Finally, when no confusion is possible, a single character *x* is used to denote a pair *(x,x)* where the lexical character is realized as itself on the surface (other notational shortcuts exist, but we will restrict ourselves to these for now).

The <*cp*> is the *correspondence part* of the two-level rule, i.e. the part that specifies the particular pair that the rule is concerned with, and <*lc*> and <*rc*> are the *left* and *right contexts* respectively, i.e. the parts that specify the context within which the rule applies. The <*operator*> specifies the *type* of the rule, as described below.

*Context restriction rules*, which take the form

<*cp*> $\Rightarrow$ <*lc*> — <*rc*>

specify a context <*lc*> — <*rc*> within which correspondence <*cp*> is *allowed* to take place. In other words, if <*cp*> is *(x,y)* and *x* appears in a lexical form surrounded by <*lc*> on the left and <*rc*> on the right, then it *may* realize as *y* on the surface (we say that some pair *(x,y)* is *surrounded* by <*lc*> and <*rc*> if whatever appears immediately to the left and right of *(x,y)* is matched by regular expressions <*lc*> and <*rc*> respectively). In general, for a pair of lexical and surface forms to be orthographically correct, every one of its individual lex-

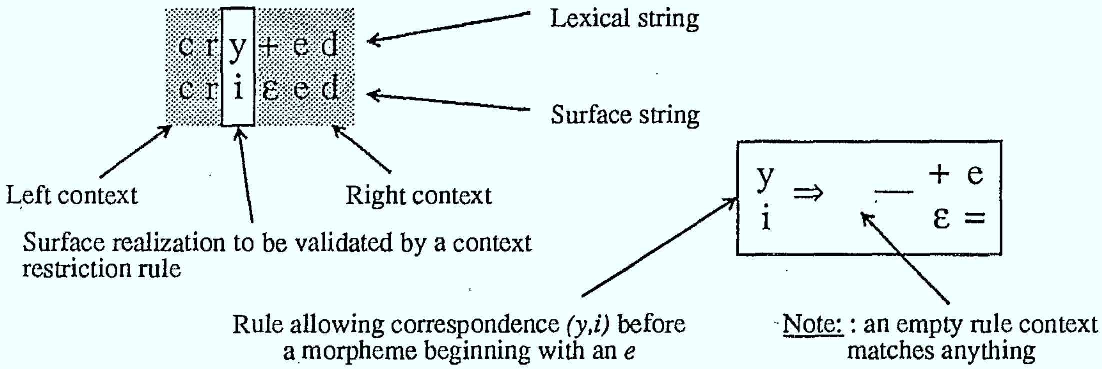


**figure 2** Pairofcorrespondinglexicalandsurfaceforms,andmatching context restriction rule.

ical/surface character pairs must be validated by a context restriction rule. This is illustrated in figure 2. Notice how CPSs are written vertically, with the lexical character on top: this is the standard notation (but it doesn't fit that well in written text, so we alternate between the two notations).

*Surface coercion rules* take the form

*<cp>* ⇐ *<lc>* — *<rc>*

and specify the context within which a correspondence *<cp>* is *forced* to take place: if *<cp>* is *(x,y)*, and *x* appears within a lexical string surrounded by *<lc>* and *<rc>*, then it *must* realize as *y* on the surface. For a particular lexical/surface forms pair to be orthographically correct, none of its individual character pairs must contradict any of the surface coercion rules. What is understood by a 'contradiction' is illustrated in figure 3. Note that the existence of a surface coercion rule to force some correspondence *(x,y)* does not imply that this correspondence is *allowed*. What such rules say is something like "No matter what *x* is normally allowed to realize as, in this context, it must *not* realize as anything other than *y*". As a result, this type of rule turns out to be useful mostly to specify *disallowed* correspondences, as in the rule:

$$\begin{matrix} a \\ \text{-}b \end{matrix} \Leftarrow c — d$$

which reads "Surrounded by *c* and *d*, lexical character *a* must realize as a character that is not equal to *b* (*a may not* realize as *b*)". Therefore, such a rule forces a given correspondence *not* to take place. It is then a matter of checking matching context restriction rules to find what *a* may realize as in the context.

*Composite rules* are used to specify correspondences that are both allowed and forced. They are obtained by combining a context restriction rule and a surface coercion rule that share the same correspondence part and left and right contexts:

*<cp>* ⇔ *<lc>* — *<rc>*

is equivalent to

*<cp>* ⇒ *<lc>* — *<rc>* and

*<cp>* ⇐ *<lc>* — *<rc>*.

Full descriptions of the orthography of a language are, in principle, collections of

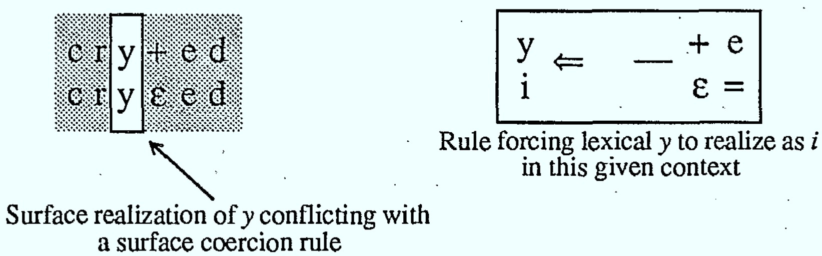


**figure 3** Example of lexical/surface pair ruled out by a surface coercion rule

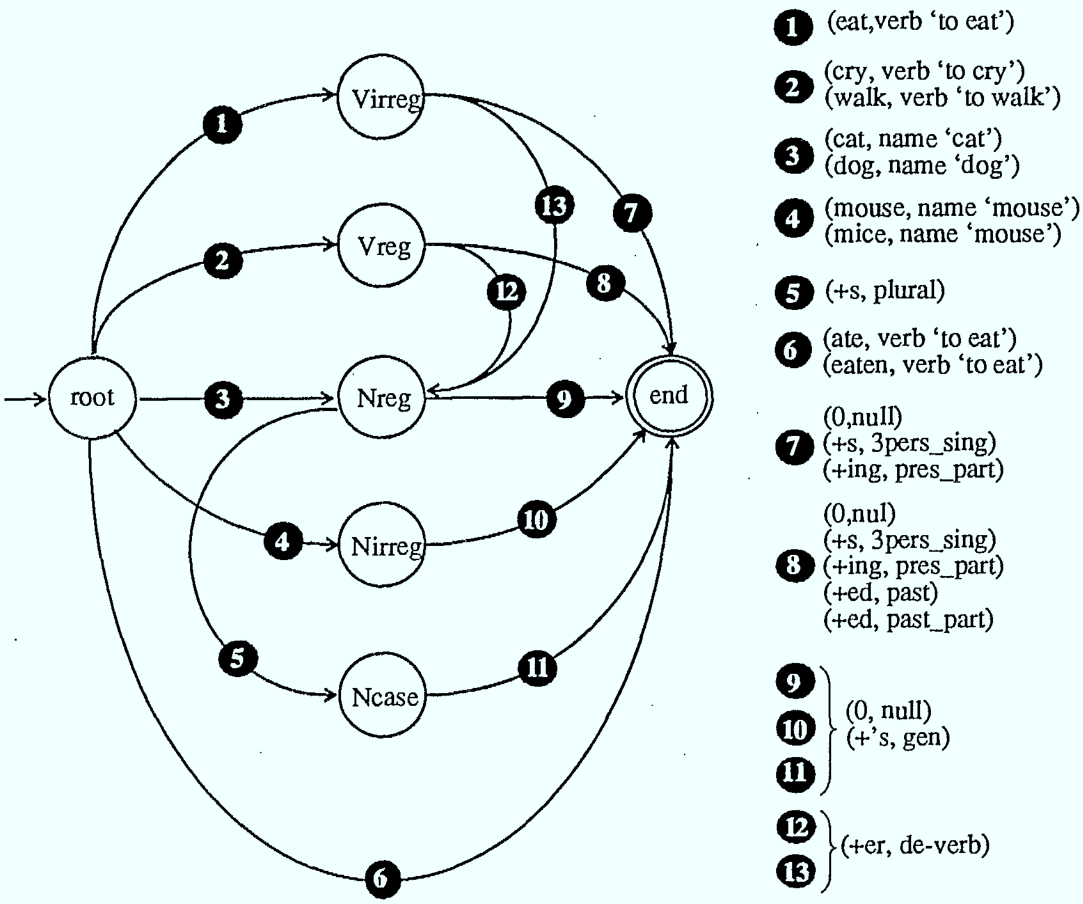


**figure 4** Transducer corresponding to lexicon of figure 1.

two-level rules. For a given character pair, all context restriction rules are taken *disjunctively* (*one* of them must be satisfied, i.e. match the current character and context), and this disjunction is then taken *conjunctively* with all surface coercion rules (*all* of them must be satisfied, i.e. not be contradicted if they match the current context and lexical character). For a pair of surface and lexical forms to be accepted by the set of rules, this logical statement must be satisfied at every position in the string. Since the rules simply state relations between the two levels of representation, the formalism is not biased towards analysis or synthesis, and may be used for both.

## 3 The Implementation

The key idea behind the implementation of Koskenniemi's model is that both its formalisms (lexicon and two-level rules) define regular languages, for which efficient parsers are relatively easy to construct: individual finite-state devices corresponding to the lexicon and two-level rules may be produced and then used in parallel to analyze or generate surface forms. How this is done is discussed below.

A *finite-state transducer* is a special type of finite-state automaton that recognizes a

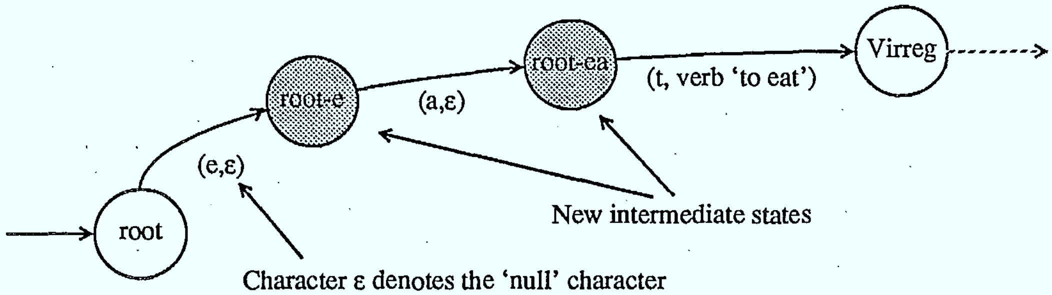


**figure 5** Expansion of a transition of the lexical transducer.

language over an alphabet of pairs of characters. In formal terms, if $A$ and $B$ are alphabets, then an automaton that recognizes some language over $(A \times B)^*$ is a transducer. This type of computational device is interesting for us because it allows to describe relations between strings over different alphabets, insofar as the computation of these relations only requires a finite amount of memory. As it turns out, this appears to be the case for the relations that exist between lexical descriptions and lexical forms, and between lexical and surface forms.

If we take a particular lexicon under Koskenniemi's formalism, and consider the set of all morphemes as an alphabet (i.e. every morpheme of the lexicon is a 'symbol' in the alphabet), then it is obvious that the morphological rules, as encoded with continuation classes, define a regular language over this alphabet. Similarly, if lexical descriptions of words are obtained by concatenating descriptions of individual morphemes, then the morphological rules also define a regular language over the alphabet of descriptions attached to morphemes. From thereon, it is easy to see how a transducer that recognizes precisely all lexically well-formed morpheme/description pairs may be constructed. Such a device corresponding to the lexicon of figure 1 appears in figure 4 (drawing conventions of [HU79] for finite-state devices are used throughout this text). It is then a trivial matter to transform this transducer into one that operates on lexical forms instead of morphemes (see figure 5), so that we obtain an efficient computational device to recognize well-formed pairs of lexical descriptions and lexical forms.

Of course, we are interested in doing more than simply recognize well-formed pairs: we also want to perform *analysis* and *synthesis* operations. In formal terms, if we have a transducer $M$ recognizing some language $L(M)$ over $(A \times B)^*$, then an *analysis* of some string $s$ of $A^*$ is some other string $t$ of $B^*$ such that there is a string $x$ of $L(M)$ with projection on $A$ $\Pi_A(x) = s$, and projection on $B$ $\Pi_B(x) = t$ (*synthesis* is defined in a symmetrical fashion). Efficient graph-searching methods exist to compute such functions with transducers, and we will not discuss these here.

Two-level orthographical rules define a language over the alphabet of pairs of lexical and surface characters, which may also be recognized by transducers, although their construction is not as obvious as that of the lexicon's. As a matter of fact, the method by which these devices are produced from sets of two-level rules is a complex one for which, to our knowledge, no clear and formal description exists. Essentially, a set of two-level rules may be partitioned into subsets of rules sharing the same correspondence part (<*cp*> part), and

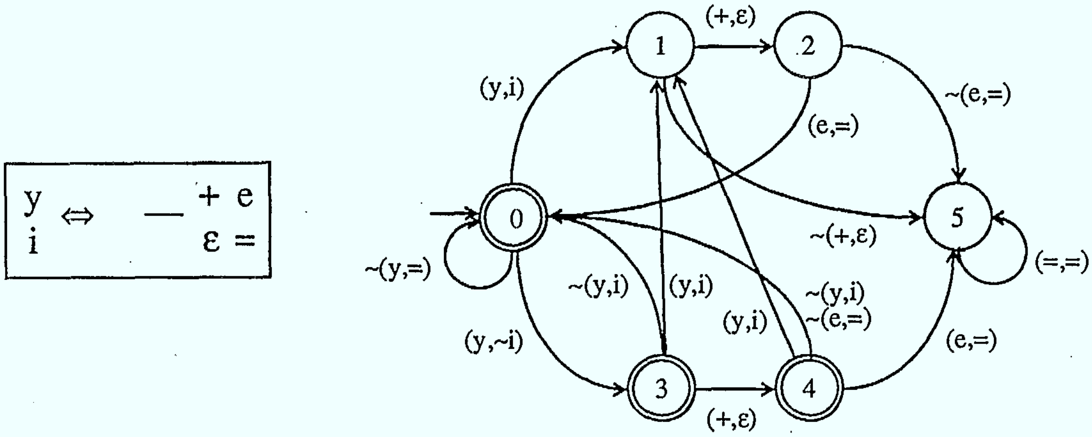


**figure 6** Two-level composite rule, and corresponding transducer.

for each of these subsets, a transducer may be constructed such that a string over lexical and surface pairs of characters is accepted as orthographically correct if and only if it is recognized by all transducers.

Such a transducer corresponding to the composite rule responsible for the *(y,i)* correspondence in English appears in figure 6 (accepting states are doubly circled). Here, symbol = within a pair of characters stands for "anything" (as before), and symbol ~ stands for a 'negation'. For example, the pair *(y,~i)* refers to lexical character *y* realized as anything *but i* on the surface, *~(y,i)* refers to all pairs not equal to *(y,i)*, and *~(y,=)* will match any character pair whose lexical character is not *y*, no matter what its surface character is. The transducer is constructed in such a way that starting from state *0*, any string will be accepted, except those containing an occurrence of *(y,i)* that is not immediately followed by *(+,ε)* and *(e,=)*, and those containing a lexical *y* not realized as *i* that is immediately followed by *(+,ε)* and *(e,=)*. This reflects the intention of the rule, which says that *y must* realize as *i* in the given context, and that as it is the only rule concerning correspondence *(y,i)*, it is also the only context within which it *may* appear. If there was another rule concerning this correspondence, then *both* rules would have to be implemented within the same unique transducer.

Actually, since regular sets are closed under intersection, it is possible to combine all transducers obtained from a set of two-level rules into a single one. In theory, this produces a finite-state device whose number of states is a multiplicative factor of the number of states of the individual components from which it was obtained. However, the nature of the 'orthographical problem' is such that the resulting transducer may usually be optimized into one with a number of states closer to the sum than to the product, which makes the operation interesting if efficiency is a concern.

So ultimately, if we combine transducers as described above, we end up with just two transducers: one defining a language over pairs of lexical and surface characters (the 'orthographical language'), and one over pairs of lexical descriptions and lexical forms (the 'morphological language'). How these devices interact with one another varies with imple-

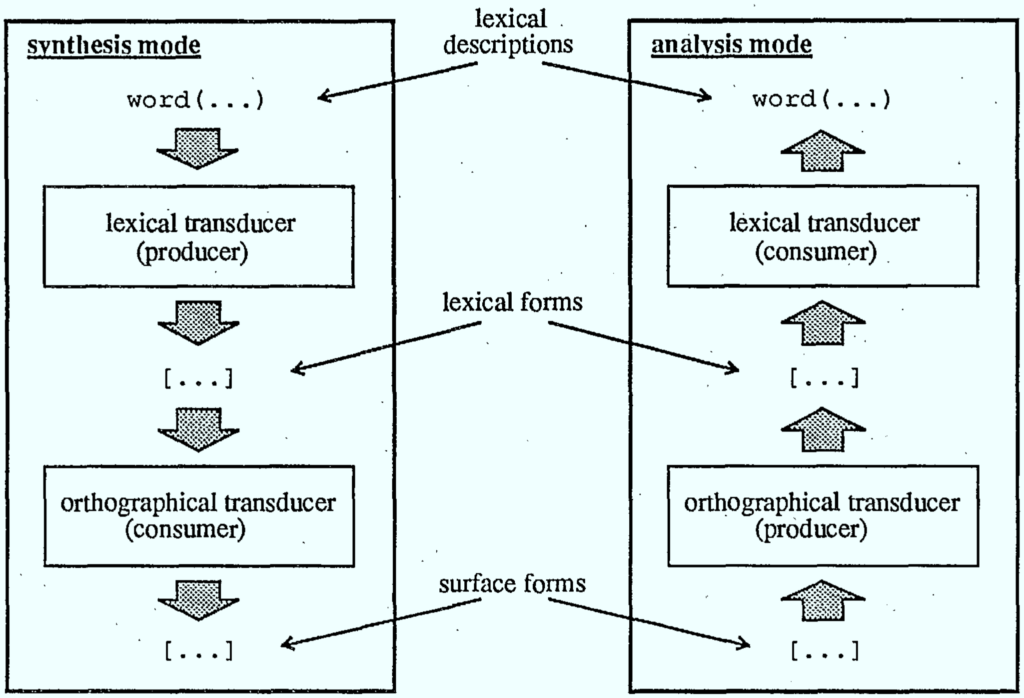


**figure 7** Producer-consumer relationships between orthographical and lexical transducers.

mentations, but we can imagine some sort of 'consumer-producer' relationship to exist between the two (figure 7), so that one 'consumes' the lexical string as it is 'produced' by the other (which one is 'producer' and which one is 'consumer' is determined by whether the system is working in analysis or synthesis mode). The resulting system is a left-to-right parser which, depending of course on the exact nature of the lexicon and morphological and orthographical rules, can turn out to be quite efficient, both in analysis and synthesis.

## 4 Other Approaches

Being one of the few computational models for morphology and orthography to be of some interest from a linguist's point of view, and also the best known, Koskenniemi's two-level model has been the object of various criticisms. Let us have a look at what the major subjects of disagreement are, and at some of the alternative solutions proposed.

- *Encoding of Morphological Rules:* The two-level model is often criticized for the 'weakness' of its encoding of morphological rules: although not proven insufficient, it appears to be all too often inadequate to describe morphological phenomena other than suffixations and prefixations. Dolan ([Dol88]) lists such things as *reduplications* (repetition of a morpheme within a word), *infixations* (insertion of a morpheme 'within' another one) and *circumfixations* ('surrounding' of a morpheme by another one), while Anderson ([And88]) and Boisen ([Boi88a]) mention 'non-affixal' phenomena (e.g. 'ablaut' relations in some English verbs: *sing*, *sang*, *sung*) as other sources of problems. Typically, encoding these within the two-level formalism will require a lot of *lexicalization* (e.g. entering all forms of the verb *to sing* as distinct morphemes in the lexicon) and reduplication of information (e.g. denoting the optional combination of elements of some sublexicon with a 'circumfix' requires the duplication of that whole sublexicon).

This has been improved in later presentations and implementations of the model, notably by providing more sophisticated methods of encoding and constructing lexical descriptions (feature structures, etc.; see [DKK87]), but many suggest that the problem comes mainly from the fact that continuation classes form a regular 'skeleton', and opt for a 'phrase structure' rule formalism, an idea first suggested by Karttunen & Wittenburg [KW83]. For example, Bear describes morphological rules using a PATR-type of formalism ([Bea86]), and Boisen's formalism is based on DCGs ([Boi88a]).

- *Parallel Application of Orthographical Rules*: Traditional models of generative phonology describe phonological phenomena using rewrite rules, meant to be applied 'in cascade', i.e. each rule applying on the output of the preceding one, and feeding its output to the next. Correspondences between the surface form and the underlying lexical form are then obtained by passing through several layers of intermediate representations, corresponding to the state of affairs between applications of rules. One hypothesis at the basis of Koskenniemi's model is that all intermediate levels between the surface and lexical forms may be bypassed, and correspondences between the two levels described directly with rules that apply simultaneously.

This view is not shared by all, and some models still favor the older method. Hankamer for example, suggests a parsing method for Turkish which, while being very close to that of Koskenniemi in its encoding of morphological rules, advocates a cascade application of orthographical rules (see [Han86]).

- *Separation between Orthography and Morphology:* One of the most important principles underlying Koskenniemi's model is the assumed separation between orthographical and morphological affairs: all that orthographical rules should know about higher levels (morphology, syntax, etc.) is communicated through the intermediate lexical representation, and vice-versa. This view is often challenged, as can be seen from the following examples.

In a model that is otherwise very close to Koskenniemi's, Bear ([Bea88]) suggests attaching what he calls 'negative rule features' to morphemes of the lexicon. What these features actually do is 'locally disable' certain orthographical rules so as to prevent them from being applied when 'exceptions' occur. This means that the morphological component has direct control on the behavior of the orthographical component. Note that this does not produce a formalism more powerful than Koskenniemi's: using negative rule features is absolutely equivalent to putting annotations in the lexical string. However, it does provide the advantage of not having to consider exceptions when writing orthographical rules, but only when writing the lexical entries to which these exceptions apply.

In a similar vein, Hankamer ([Han86]) uses different sets of rules for *roots* and for *suffixes*, thus assuming that orthographical rules 'know' what type of morpheme they are dealing with. This is actually just another way of locally disabling selected rules, only more restricted than Bear's.

Cornell's IceParse model of inflectional parsing ([Cor88]) is a clearer departure from Koskenniemi's approach: in his view, morphology may be seen as some sort of 'orthographical phenomena' (as in e.g. 'umlaut' and 'ablaut' relations). Therefore, suffixes do not have corresponding lexical entries in his model, morphological rules are denoted the same way orthographical rules are, and define relations between the same two levels of representation, i.e. the surface form and some underlying representation akin to Koskenniemi's lexical form. The difference between the two types of rules is that morphological rules have the ability to affect the lexical descriptions of the lexemes to which they are applied. At the implementation level, both types of rules are translated into finite-state transducers.

At the other end of the spectrum lies Boisen's DCM formalism, where orthographical rules, encoded as transformational rules, are applied 'within' morphological rules. In other words, the application of a particular orthographical rule, in Boisen's view, is always a consequence of the application of a specific morphological rule.

## 5 Using Koskenniemi's Formalism in a Continuous-text Environment.

As mentioned earlier, when considering the integration of a formalism such as Koskenniemi's to a natural-language processing system, we are faced with the following situation: while the two-level model was originally designed to deal with single isolated words, we want it to work in a environment of continuous-text, i.e. one where borderlines between words are not always clearly marked. For example, while in most written languages the normal boundary character between words is the 'space', it is likely that things like *British Columbia* and *Department of Agriculture* will be viewed as words of their own, not to be decomposed in any way. This means that in addition to recognizing words, we have to deal with the problem of *isolating* them in the text.

In a single word context, a program implementing the two-level model is normally asked questions of the type "Give a lexical description of the following word if one exists" (the 'analysis question') or "If possible, produce a word having this lexical description" (the 'synthesis question'). It seems quite obvious to us that a program capable of answering these questions, provided with just a little extra information, may be used to answer questions about *sentences*, such as "Find all words of the language that match the left-hand side of this sentence, along with their corresponding lexical descriptions", or "Produce a word with this lexical description at that position of the sentence". Here, the word 'sentence' should be taken to mean simply 'list of words', and what the 'little extra information' required actually corresponds to is a description of how words appear within sentences (what the word boundaries look like, etc.).

Our claim is that this information about sentences may be encoded as *morphological* and *orthographical* information. This may be done by forcing the appearance of an explicit 'word-boundary character' at the end of each word of the language — a trivial matter under the two-level formalism's encoding of morphological rules if this character is seen as a morpheme — and by controlling its surface realization 'orthographically' with two-level

(a) $\genfrac{}{}{0pt}{}{\wedge}{\_} \Rightarrow \text{—}$

(b) $\genfrac{}{}{0pt}{}{\wedge}{\varepsilon} \Leftrightarrow \text{—}\ P$

*P* a punctuation character

(c) $\genfrac{}{}{0pt}{}{\wedge}{\varepsilon} \Leftrightarrow A\ \text{—}$

*A* the apostrophe'

**figure 8** Two-level rules sanctioning the realizations of word-boundary character '^'.

rules. Without going into too much detail, we can see how this may be done: let us take character '^' to be the *word-boundary* symbol. In general, it will realize on the surface as a space (that we will denote by '_'). However, there is a certain number of situations where such space must not appear, for example before a punctuation character, or after an apostrophe. This may be taken care of by a set of two-level rules, as shown in figure 8. Rule (a) says that '^' may always realize as a space '_' on the surface. However, other rules specify situations where it may and must realize as the null character ε: before a punctuation (rule b) and after an apostrophe (rule c).

One thing that should be noted is that this will work only if orthographical rules are applied to *whole sentences*, rather than to individual words: in general, the realization of a word boundary will be conditioned by the appearance of both what precedes it and what follows it. When integrating the two-level formalism to a natural-language processing system, one of our major concerns will be to allow for this 'global' application of rules.

Interestingly, it seems that besides standard 'intra-word' orthography, the surface appearance of word-boundaries is not the only thing that may be treated orthographically if two-level rules are applied in this way. For example such things as *elisions* and *contractions* could also be labeled as 'inter-word orthographic phenomena'. Let us have a quick look at this: in French, a good number of words take a different form whether they appear in front of a vowel (or mute *h*) or a consonant. For example, the *e* of word *le* (definite article or pronoun) disappears and is replaced by an apostrophe (') before a vowel, as in *l'avion* (a similar thing occurs in English with indefinite article *a*). This phenomenon is hard to describe syntactically, because it does not seem to depend in any way on the syntactic relation between the elided word (in this case, *le*) and the word which caused the elision.

The two-level rules of figure 9 solve the problem, not only for *le* but also for a bunch of others such as *ne*, *se*, *me*, *te*, etc., which behave similarly. Rule (a) says that lexical *e* may and must realize as an apostrophe only within words such as these appearing in front of a vowel or an *h*. Otherwise, it may always realize as itself (rule b).

A contraction occurs when the juxtaposition of two distinct words results in their realization as a single surface form. There are examples of both compulsory contractions (in French, preposition *à* and article *le* always contract to *au*) and of optional contractions (in

(a) $$\genfrac{}{}{0pt}{}{e}{'} \Leftrightarrow \genfrac{}{}{0pt}{}{\wedge}{\varepsilon} C \;—\; \genfrac{}{}{0pt}{}{\wedge}{\varepsilon} V$$

$$C \in \{c, d, j, l, m, n, s, t\}$$
$$V \in \{a, e, h, i, o, u, y\}$$

(b) $$\genfrac{}{}{0pt}{}{e}{e} \Rightarrow \;—$$

**figure 9** Two-level rules controlling elisions in such French words as *ce, de, je,* etc.

English, *do* and *not* may contract to *don't*). Once again, this type of phenomena does not seem to lend itself very well to syntactic description. The set of rules in figure 10 describes the compulsory contraction of *à le* into *au*: rules (a) through (e) sanction one by one the appearance of the characters within the contraction of *à le* into *au*. Notice that in order for this to work, preposition *à* must be given a lexical form *à*υ, where υ is a lexical character which realizes as *u* within the contraction , but as ε otherwise (rule f).

Of course, the rules discussed above should always be considered as part of a larger set, which together constitutes the orthographical component. We can probably imagine starting with a set of rules describing intra-word orthography, modifying these so as to prevent them from leaping across word-boundaries (intra- and inter-word phenomena are usually distinct), and then adding on the inter-word rules (within which word-boundary markers should appear explicitly, for similar reasons). It should also be pointed out that the appearance of the lexical representation we have seen so far reflects more our desire to keep the discussion at an intuitive level than true linguistic accuracy.

It is interesting to note that while two-level rules are able to handle all three of these phenomena (word-boundary realizations, elisions and contractions), they appear to be rather ill-suited for the description of the last type (contractions). What this suggests is that although intra-word orthography *is* regular (it may be described adequately by finite-state devices), there may exist formalisms better fit to describe it than two-level rules. This idea is further supported by the fact that the set of rules required to describe the contraction of *à le*

(a) $$\genfrac{}{}{0pt}{}{\grave{a}}{a} \Leftrightarrow \genfrac{}{}{0pt}{}{\wedge}{=} \;—\; \genfrac{}{}{0pt}{}{\upsilon\ \wedge\ l\ e\ \wedge}{u\ \varepsilon\ \varepsilon\ \varepsilon\ =}$$

(b) $$\genfrac{}{}{0pt}{}{\upsilon}{u} \Leftrightarrow \genfrac{}{}{0pt}{}{\wedge\ \grave{a}}{=\ a} \;—\; \genfrac{}{}{0pt}{}{\wedge\ l\ e\ \wedge}{\varepsilon\ \varepsilon\ \varepsilon\ =}$$

(c) $$\genfrac{}{}{0pt}{}{\wedge}{\varepsilon} \Leftrightarrow \genfrac{}{}{0pt}{}{\wedge\ \grave{a}\ \upsilon}{=\ a\ u} \;—\; \genfrac{}{}{0pt}{}{l\ e\ \wedge}{\varepsilon\ \varepsilon\ =}$$

(d) $$\genfrac{}{}{0pt}{}{\grave{a}}{a} \Leftrightarrow \genfrac{}{}{0pt}{}{\wedge\ \grave{a}\ \upsilon\ \wedge}{=\ a\ u\ \varepsilon} \;—\; \genfrac{}{}{0pt}{}{e\ \wedge}{\varepsilon\ =}$$

(e) $$\genfrac{}{}{0pt}{}{\grave{a}}{a} \Leftrightarrow \genfrac{}{}{0pt}{}{\wedge\ \grave{a}\ \upsilon\ \wedge\ l}{=\ a\ u\ \varepsilon\ \varepsilon} \;—\; \genfrac{}{}{0pt}{}{\wedge}{=}$$

(f) $$\genfrac{}{}{0pt}{}{\upsilon}{\varepsilon} \Rightarrow \;—$$

**figure 10** Two-level rules in charge of the contraction of *à le* into *au.*

into *au*, it seems, could be combined in a unique and fairly simple transducer. We leave open the question of what such an alternative formalism could look like.

Another problem that appears in the course of designing orthographical rules for a language stems from the fact that some of these rules appear to be inherently ambiguous, and to generate multiple hypotheses both in analysis and in synthesis. For example, in French, surface form *des* may be analyzed either as plural form of indefinite article *un*, or as the contraction of preposition *de* and plural definite article *les*. On the other hand, the latter lexical string (juxtaposition *de+les* of the lexical forms of *de* and *les*) may produce either the contraction *des* if *les* is an article, or simply *de les* if *les* is a pronoun.

In a system such as the one we want to build, surface forms yielding multiple lexical forms are not considered a problem, because higher levels (the lexicon, morphological rules, syntactic rules, etc.) should be able to rule out unacceptable lexical forms. However, ambiguities in the other direction are problematic: for example, if the orthographical component (i.e. the component in charge of applying orthographical rules in an implementation of the model) does not 'know' if a particular instance of lexical form *les* corresponds to the article or to the pronoun, then it has no way of deciding whether a contraction with *de* must take place or not. Obviously, orthographical rules cannot work properly without this knowledge of "which rule applies in what situation". The problem, it seems, is that not all of it may reside within the orthographical component itself. If a parallel may be drawn between orthography in written text and phonology in spoken language, then the following example should be sufficient to illustrate our point. In the French phrase "*un marchand de draps anglais*", the presence of a liaison between *draps* and *anglais* depends on the scoping of adjective *anglais* (whether it applies to *marchand* or to *draps*. Well, yes, this is a bit far-fetched...). In this example, the application of a specific phonological rule is conditioned not by the immediate phonological environment, but by the underlying syntactic structure, and may even have deeper 'roots'.

So it seems reasonable to assume that not all of the knowledge relevant to orthography resides in the orthographical component itself, and that we may talk of *lexically*, *morphologically* or even *syntactically conditioned orthography*. Therefore, when orthographical ambiguities arise, the orthographical component 'knows' how to resolve it only if it is actually 'told' by some higher level component. It seems that the simplest way of modeling this transfer of knowledge in our system is through direct annotations on the lexical string. In other words, we can imagine that the lexical representation actually contains much more information than the appearance of the surface would suggest. We have already mentioned morpheme-boundary and word-boundary characters, but we could probably think of many other objects appearing in the lexical string that do have surface realizations, although not always obvious. For example, we could have the syntax (or the lexicon) to insert explicit *pauses* at certain selected positions of a sentence (or of a word) to locally inhibit some or all orthographical rules. A comma could be the surface realization of such a 'strong' pause, but 'weaker' pauses could also exist that have more subtle orthographical effects, such as inhibit the 'elision rules' in front of some words beginning with an *h*, or the 'contraction rule' of *de les* when *les* is a pronoun, etc. Other types of markers could be imagined to exist. The important thing is to see these as a way of communicating higher-level information to the orthographical component.

Of course, lexical entries, morphological rules and even grammar rules must be written with orthography in mind, and orthographical rules must take into account the additional information, but this seems manageable. In any case, we leave open the question of how exactly this may be done.

## Conclusion

The first part of this paper was a presentation of existing computational models of morphology. Many current approaches are derived from Koskenniemi's "two-level" model, so this was examined in detail first, followed by a quick survey of some of the alternative solutions proposed. The second part discussed how Koskenniemi's model could be used in a continuous-text environment to deal with "inter-word" phenomena, such as elisions and contractions.

This last topic was only briefly touched, and several questions were left open, in particular on whether the two-level model was actually the most appropriate to describe some of these phenomena. Finding a formalism better fit to this task, as well as exploring communication schemes between the orthographical component and higher levels of linguistic processing both seem to us interesting topics for future research.

In spite of all its deficiencies, we favour using Koskenniemi's model as a starting point for such research work, this for a number of practical reasons: first, there are many natural languages for the description of which the two-level formalism turns out to be quite sufficient, and to produce efficient parsers. More importantly, the model is not biased toward analysis or synthesis, and may be used for both. In our view, this is a major advantage over approaches such as those of Hankamer, Boisen, Cornell and Dolan to name but a few, which are all designed primarily to perform analyses. This is not just a convenient computational property: it is also a strong criterion of linguistic accuracy. Finally, Koskenniemi's model is, in some way, *representative* of current approaches in computational morphology. This property is important to us, because it confers a certain level of generality to our results: we believe that the ideas presented in section 5 may be applied to most finite-state morphological models.

## References

[And88] Anderson, Stephen R. (1988), Morphology as a Parsing Problem, in *Morphology as a Computational Problem*, UCLA Occasional Papers #7: Working Papers in Morphology, Dept. of Linguistics, UCLA.

[Bar86] Barton, Jr., G. Edward (1986), *The Computational Complexity of 2-level Morphology*, AI Memo 856, MIT AI Laboratory.

[Bea86] Bear, John (1986), A Morphological Recognizer with Syntactic and Phonological Rules, *COLING 86* (*Proceedings of the 11th International Conference on Computational Linguistics*), pp. 272-276.

[Bea88] Bear, John (1988), Morphology with Two-level Rules and Negative Rule Features, *COLING 88*, pp. 28-31.

[Boi88a] Boisen, Sean (1988), Parsing Morphology using Definite Clauses, in *Morphology as a Computational Problem*, UCLA Occasional Papers #7: Working Papers in Morphology, Dept. of Linguistics, UCLA.

[Boi88b] Boisen, Sean (1988), Pro-KIMMO: a Prolog Implementation of Two-level Morphology, in *Morphology as a Computational Problem*, UCLA Occasional Papers #7: Working Papers in Morphology, Dept. of Linguistics, UCLA.

[Cor88] Cornell, Thomas L. (1988), IceParse: A Model of Inflectional Parsing and Word Recognition for Icelandic Ablauting Verbs, in *Morphology as a Computational Problem*, UCLA Occasional Papers #7: Working Papers in Morphology, Dept. of Linguistics, UCLA.

[DKK87] Dalrymple, Mary, Ronald M. Kaplan, Lauri Karttunen, Kimmo Koskenniemi, Sami Shaio and Michael Wescoat (1987), *Tools for Morphological Analysis*, Report No. CSLI-87-108, Stanford University CSLI.

[Dol88] Dolan, William B. (1988), A Syllable-based Parallel Processing Model for Parsing Indonesian Morphology, in *Morphology as a Computational Problem*, UCLA Occasional Papers #7: Working Papers in Morphology, Dept. of Linguistics, UCLA.

[Han86] Hankamer, Jorge (1986), Finite-state Morphology and Left-to-right Phonology, *West Coast Conference on Formal Linguistics*, 5, pp. 41-52.

[HU79] Hopcroft, John E. and Jeffrey D. Ullman (1979), *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley.

[Kar83] Karttunen, Lauri (1983) KIMMO: A General Morphological Processor, *Texas Linguistic Forum #22*, pp. 165-186.

[Kos83] Koskenniemi, Kimmo (1983), *Two-level Morphology: a General Computational Model for Word-form Recognition and Production*, Publication 11, Dept. of General Linguistics, University of Helsinki.

[KW83] Karttunen, Lauri and Kent Wittenberg (1983), Two-level Morphological Analysis of English, *Texas Linguistic Forum #22*, pp. 217-228.

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**Pour plus de détails,**
**veuillez communiquer avec :**

*Le Centre canadien de recherche*
*sur l'informatisation du travail*
1575, boulevard Chomedey
Laval (Québec)
H7V 2X2
(514) 682-3400

**For more information,**
**please contact:**

*Canadian Workplace*
*Automation Research Centre*
1575 Chomedey Blvd.
Laval, Quebec
H7V 2X2
(514) 682-3400