3-8-1991

Morphogen: A Morphology Grammar Builder and Dictionary Interface Tool

Joseph E. Pentheroudakis
Dan W. Higinbotham

Follow this and additional works at: https://scholarsarchive.byu.edu/dlls

BYU ScholarsArchive Citation
Available at: https://scholarsarchive.byu.edu/dlls/vol17/iss1/9

This Article is brought to you for free and open access by the All Journals at BYU ScholarsArchive. It has been accepted for inclusion in Deseret Language and Linguistic Society Symposium by an authorized editor of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen amatangelo@byu.edu.
INTRODUCTION

MORFOGEN is a finite state compiler, taking as input text files containing inflectional and derivational paradigms and compiling them into a finite state machine. The paradigms are tables specifying the allowable sequences of morphemes in the language as well as the forms of the morphemes proper to each inflectional class. These paradigms are assigned unique names, identifying the inflection classes in the language under analysis and used when marking the words in the lexicon.

MORFOGEN also includes a morphological analyzer, which accesses the compiled rules to identify the morphemes in an inflected string and suggest one (or several) base forms. These forms are then looked up in the lexicon using dictionary access routines, which can be customized to interface with any on-line dictionary. If the lexicon contains inflectional class information, that information can then be used to accept or reject the analyses suggested by the analyzer.

The purpose of this paper is to present an overview of the functionality of MORFOGEN, and to describe MORFOGEN’s rule formalism and its linguistic motivation. The design of MORFOGEN’s high-level language and finite state compiler was driven by the need for a morphological analysis tool able to express (and test) linguistic phenomena clearly and efficiently, without sacrificing expressive power to the requirements of the underlying implementation. Since we will not be describing the implementation of the finite state automaton, but rather a high-level formalism for the description of morphological systems, a direct comparison with the two-level approach [Koskiennemi 1984; Antworth 1990] is not within the purview of this paper. We should point out, however, that a central difference between two-level systems and MORFOGEN lies in the basic units of description and their organization. Two-level rules are finite-state automata based on the correspondences between surface forms and lexical (or underlying) forms; the basic units of description in MORFOGEN rules are abstract morpheme categories and their organization in paradigms. Some of the problems often mentioned in connection with the two-level model, namely, the awkwardness and inefficiency of expressing nonconcatenative processes and the need to hand-compile the finite state tables, have been addressed in the design of MORFOGEN. (For an overview of the history and implementations of two-level systems, see [Antworth 1990]; research in progress may result in the development of a compiler for PC-KIMMO [Antworth, personal communication].)

This paper is organized as follows. Section 1 is the introduction. Section 2 presents MORFOGEN’s approach to the description of morphological systems, illustrating it with examples from French and Turkish. Section 3 describes and illustrates the lower-level morphological operators used to parse word forms. Section 4 is a discussion of the priority levels implemented in MORFOGEN, used in grouping paradigms in terms of their frequency in the language. Section 5 briefly discusses the contribution of the lexicon in the analysis process. MORFOGEN applications and platforms are presented in section 6. Section 7 is a conclusion and a brief discussion of future work in extending MORFOGEN’s functionality.

MORFOGEN DESIGN

MORFOGEN rules encode morphotactic information, specifying allowable sequences of morphemes, and allomorphic variation, providing for the description of allomorphs and the distribution of individual allomorphs in each inflectional class. The
units in a MORFOGEN analysis are morpheme categories, the members of those categories, and, on the lowest level, the descriptions of the (orthographic) forms of those elements. Even though MORFOGEN descriptions make reference to morphemes and allomorphs, however, in terms of their organization they reflect the insights of the Word and Paradigm model of morphology [Matthews 1974]: MORFOGEN rules are paradigms, or groups of tables identifying the forms defining the inflectional patterns in a language. Words belonging to a given paradigm will inflect like the model for that paradigm listed in the rule file; the paradigm name can be stored in the lexicon to help the analyzer reject spurious analyses and to constrain derivations.

An important feature of MORFOGEN is that it allows an analysis of the inflectional and derivational morphology of a language which is independent of the identification of the actual portion(s) of the inflected forms associated with a particular morpheme category. Thus, the form ‘children’ can be analyzed either as ‘child+ren(PLUR)’ or simply as ‘children (PLUR)’; however, regardless of the process used to identify the base form, both analyses will identify ‘children’ as ‘child + PLURAL’. Whether the plural morpheme is identified with the characters ‘ren’ or whether the entire form ‘children’ is analyzed as plural is, in a sense, immaterial in the description of paradigms, especially in the case of paradigms represented by very few members. The actual morphological operations used to identify the form of a morpheme can be expressed without disturbing the overall description of the morphological system of a language.

The expression of such morphological operations is accomplished by means of several specially defined operators which can handle both concatenative and non-concatenative phenomena which range from simple suffixation and prefixation to operations involving discontinuous morphemes, infixation, gemination, degemination, reduplication etc. Morphological operations can be assigned a morph name, and their distribution and cooccurrence restrictions can be stated using that name.

Finally, the content of morphemes can also be specified in the MORFOGEN rule file; this information can then be used by a parser and can help determine the syntactic or semantic function of the inflected form. Furthermore, the relative order in which morphemes are identified will be reflected in the derivation history returned by the analyzer, and will be available to the application within which the analyzer is integrated; thus, the two possible bracketings for a form like ‘unmasked’, namely, [un+[mask+ed]] (‘not masked’) and [[un+mask]+ed] ‘unmask+past’, will differ in the relative nesting of the morphemes ‘un’ and ‘ed’.

A Sample French Morphology: Finite Verb Forms

Consider the French verbal system, which needs to account for a total of 45 finite forms and 5 non-finite forms. The 45 finite forms are traditionally organized in tables representing a combined tense/mood category (cf. future indicative, present subjunctive etc.), each table in turn containing forms representing the members of a combined person/number category; the 5 non-finite forms (four participial forms marked for gender and number, and the infinitive form, identical to the dictionary form) also need to be recognized. The finite forms are typically analyzed as involving fused morphemes, each encoding a constellation of linguistic categories (tense, mood, person and number). This organization is reflected in the structure of verb conjugation tables in traditional French textbooks. Superordinate to this organization is the partitioning of the French verbal system in inflectional classes or paradigms, traditionally known as ‘conjugations’.

A MORFOGEN input file can be organized in a similar way. Partial descriptions for the CHANTER and the FINIR paradigms are shown in Fig. 1. Also included in Fig. 1 is the declaration section, normally included at the head of the input file, and identifying, among other things, the categories to be used in the analysis.

Each table in a paradigm is delimited by angle brackets, and contains two kinds of information. On the first line, it specifies the categories of the morphemes being analyzed, the name of the paradigm to which the table belongs, and whether the morpheme analyzed can be the last morpheme identified during analysis (in finite state machine terms, it indicates whether this table defines a final state). Each table references two morphemes, the
root (or inflecting morpheme), and the affix; the
table will specify the ways these morphemes
combine. The symbols used in the first line of each
table in Fig. 1 are interpreted as follows:

VERB: the root category (declared as such)
VERB-MORPH: the category of the affix attached to
VERB and analyzed in the current table
CHANTER, FINIR: the names of the paradigms
PRESENT, IMPARFAIT: additional information
about the content of the morphemes analyzed in
each table

* indicates that this is an end state in the finite state
representation, that is, that VERB-MORPH can be
the last morpheme identified during analysis

The first line of the main body in each table
specifies the operation required to derive the base
form for the string under analysis; the remaining
lines specify the operations required to identify the
word forms in the table. Each morpheme is identi-
ified at the beginning of the line by means of an
identifier, in this case the French pronouns 'je' "I",
'tu' "you", 'il' "he", 'nous' "we", 'vous' "you-pl" and
' ils' "they". These are the names that the ana-
lyzer will use when listing the morphemes that
matched during analysis. If, as shown in the exam-
ple, these names are also mapped to strings like 1-
SG (first singular), 2-PL (second plural) etc. (under
the symbol ALIASES), the analyzer will refer to the
morphemes by the latter symbols instead.

The invariant portions of the stem ('chant' and
'fin' in the example) are not interpreted literally:
rather, these act as placeholders for the stem of any
verb marked as selecting for the paradigm under
analysis, and are used in the MORFOGEN input file
to improve legibility and to allow for more efficient
development. The last character string on each line,
however, separated from the stem by a space, is
interpreted literally: this is the actual form of the
suffix which will be stripped off the inflected form.
A morpheme is successfully identified if the suffix
(or the description of a more complex operation)
matches and successfully removed from the string.

The purpose of MORFOGEN's analyzer is to
generate base forms, or dictionary forms, for
inflected strings. To generate the base form of a
French verb, an infinitive affix needs to be added to
the stem ('er' and 'ir' for the CHANTER and FINIR
paradigm, respectively). This operation is specified
on the first line of the main body of the table. During
analysis, naming a suffix in this position is inter-
preted as an instruction to add that suffix to the
stem.

Note that the actual forms of the suffixes used
here (or, more generally, the actual content of the
morphological operations used to parse an inflected
word form) do not affect the linguistic description of
the French verb form. Thus, one could analyze the
form 'finissent' as 'fini+ssen' and modify the
tables accordingly; however, the analysis 'fin+
issent' in effect introduces context into the rule,
thus ensuring that a form like 'cassent' would not be
tried here.

Fig. 1: A sample of French verb morphology
Based on these tables, a form like ‘sonnons’ “we-ring” will be analyzed as the first person plural of the present indicative of ‘sonner’ “to ring”; the analyzer returns the following response:

sonnons ->
sonner + 1-PL-PRESENT-IND

The morpheme matched is identified as 1-PL-PRESENT-IND, combining the string ‘1-PL’, the morpheme represented by its alias ‘nous’ in the table, and the morpheme name PRESENT-IND, specified on the first line of the table. This is in keeping with our analysis, which treated these forms as containing a single fused morpheme, encoding a complex set of features. As will be shown below, a parser can then use this information either indirectly, by referencing a morpheme entry by that name stored in the lexicon, or directly, by using it as an attribute name.

A Sample of French Morphology Continued: an Agglutinating Analysis

The previous example presented a fused-morpheme analysis of the finite verb forms in French; it would be possible, of course, to analyze these forms synthetically, as involving the agglutination of several morphemes. This is the approach we will use in the following analysis of French participles. A similar analysis will then be presented for Turkish, a certifiably agglutinating language.

The analysis of inflecting languages in MORFOGEN is actually a limiting case of its formalism. In the analysis of inflecting languages, each table in the paradigm identifies a sequence of only two morphemes, typically the root (or lexical category) and the category of the affix attached to the root. That this can be a final state is indicated by the asterisk at the end of the table’s header line (see Fig. 1).

Agglutination, on the other hand, typically involves several layers of morphemes. French participles, for instance, can be analyzed either as consisting of the verb stem bearing a fused morpheme encoding gender, number and verbal information, or as bearing three separate, identifiable morphemes, one for each of these categories. Let us identify three morpheme categories, GENDER, NUMBER and PARTICIPLE, each including the following morphemes:

- GENDER -> MASC
- GENDER -> FEM
- NUMBER -> SING
- NUMBER -> PLUR
- PARTICIPLE -> PRES
- PARTICIPLE -> PAST

The GENDER and NUMBER morphemes are independently motivated by the analysis of French adjectives, lending support to this analysis for French participles. The rule that combines these morphemes is:

V -> VERB PARTICIPLE GENDER NUMBER

In the MORFOGEN input file we will want to represent this sequence of morphemes, and ensure that the correct forms for each paradigm are selected. Note that in this analysis, all three morphemes are obligatory; the forms of the MASC(uline) and SING(ular) morphemes will therefore, as we will see, need to be represented by a zero morph. (An analysis not requiring zero morphs can also be written, represented by the following rule:

V -> VERB PARTICIPLE (GENDER) (NUMBER)

In this analysis, default GENDER and NUMBER features can be unified into the lexical feature structure if GENDER or NUMBER were not present in the lexical form; this can be accomplished if MORFOGEN’s analyzer is integrated into a unification-based parser, for example.)

The forms for the present and past participles of the CHANTER and FINIR classes are shown in Fig. 2; the morphemes are separated by a space, and zero morphs are shown by a 0 (zero):
Since MORFOGEN tables express arcs between two morphemes, the representation of the morphemes encoded in the participles will require three tables, one for each of the following transitions; the affix category is then linked to a table where it occupies the position of the root category:

(ROOT-CATEGORIES (VERB))
(AFFIX-CATEGORIES (PARTICIPLE GENDER NUMBER))
(ALIASES ( ANT PRES-PART
 E PAST-PART ))
(SETS ( PARTIC_ENDINGS ( i issant é ant )
 COMMON-GENDER ( null e ))

#1
< VERB PARTICIPLE CHANTER
 chant er
 ANT chant ant
 E chant é
>

< VERB PARTICIPLE FINIR
 fin ir
 ANT fin issant
 E fin i
>

< PARTICIPLE GENDER ^PARTIC_ENDINGS
 chantant/chanté
 MASC chantant/chanté %null() 
 FEM chantant /chanté e
>

< GENDER NUMBER ^COMMON-GENDER *
 chantant/chanté
 SING chantant/chanté
 PLUR chantant/chanté s
>

Fig. 3: An analysis of French participles
Since all morphemes are obligatory, the analyzer will only be allowed to consider the NUMBER morpheme as the final state; the asterisk will be appended to that table header.

This analysis is reflected in Figure 3 (see previous page).

These tables introduce several features of the MORFOGEN formalism. First, the identifier SETS in the declaration section allows a set of morphemes to be referenced by a set name. This notation is useful when linking paradigm tables. For instance, it is not sufficient to say that the VERB PARTICIPLE table can be linked to the PARTICIPLE GENDER table; we also need to specify which PARTICIPLE morphemes can be followed by which forms of the GENDER morpheme. In effect, this establishes paradigms of participle endings, defined in terms of the forms of the GENDER morphemes that they select. The participle ending paradigms are referenced as a group by specifying the name of the set that contains them (declared as PARTICIPALENDINGS) in the PARTICIPLE GENDER table. This notation allows us to collapse all PARTICIPLE GENDER transitions in one table, expressing the fact that all participle forms can be followed by the same class of GENDER morphemes.

Secondly, note the expression %#null(). This is an example of MORFOGEN’s morphological expression notation. An allomorph can be either a simple suffix or prefix, or it can be a complex operation, in the form %#NAME(operation), where NAME is an optional identifier assigned to the allomorph described by that operation. The notation %null() describes a zero morph; a null operation is specified in parentheses. (For a brief description of the class of MORFOGEN operators, see next section.) The two gender morphemes, null and e, are referenced by the set name COMMON-GENDER in the GENDER NUMBER table. Since there are probably other GENDER NUMBER tables in the rule set, to account for the gender and number morphemes in adjectival forms, specifying the set name COMMON-GENDER on the table ensures that the PARTICIPLE GENDER table will be linked to the correct GENDER NUMBER table.

This example illustrates the principles involved in determining the legal sequences of morphemes in the MORFOGEN formalism. During analysis, a table representing the state B C will be linked to a table representing the state A B only if the forms in A B are mentioned in B C, either explicitly or by means of a set name. This principle is extended to include the paradigm subcategorization information in the lexicon, and is used to accept or reject base forms: the paradigm name explicitly mentioned in the final table accessed prior to lookup must be present in the lexical entry. Schematically, the analysis of the form ‘chantantes’ “sing+PRES-PART+ FEM+PLUR” involves the following links (starting from the bottom):

Fig. 4: Analysis of the French form 'chantantes'
A Sample of Turkish Noun Morphology

The final example involves the analysis of Turkish nouns forms. Turkish can be marked for case and number, and, optionally, with a possessive marker. Turkish also has vowel harmony: the form of derivational and inflectional affixes is conditioned by the vowel quality of the stem. If we posit a HARMONY morpheme to reflect vowel harmony information, we can describe the distribution of the affix forms in terms of that morpheme (as well as in terms of the stem, of course, to account for assimilation phenomena). The structure of the Turkish noun form can then be represented as:

NOUN HARMONY (NUMBER) (POSS) CASE

To simplify this presentation and avoid positing several layers of zero morphs, we will concern ourselves only with forms including an optional POSS and an obligatory CASE marker. These categories will have the following members:

HARMONY -> öü (front, rounded)
HARMONY -> ei (front, not rounded)
HARMONY -> aI (not front, not rounded)
HARMONY -> ou (not front, rounded)
POSS -> 1-SG-POSS
POSS -> 2-SG-POSS
CASE -> ACC
CASE -> DAT

For example, the forms 'evime' "my+house+DAT", 'eve' "house+DAT", 'atInI' "your+horse+ACC" and 'atI' "horse+ACC" are analyzed as shown in Fig. 5:

<table>
<thead>
<tr>
<th>NOUN</th>
<th>HARMONY</th>
<th>POSS</th>
<th>CASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ev</td>
<td>ei</td>
<td>im</td>
<td>e</td>
</tr>
<tr>
<td>'house'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ev</td>
<td>ei</td>
<td>1-SG</td>
<td>e</td>
</tr>
<tr>
<td>'house'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>at</td>
<td>aI</td>
<td>In</td>
<td>I</td>
</tr>
<tr>
<td>'horse'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>at</td>
<td>aI</td>
<td>2-SG</td>
<td>I</td>
</tr>
<tr>
<td>'horse'</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5: Analysis of Turkish noun forms

The analysis of these forms will require the following transitions:

```
NOUN HARMONY
   HARMONY CASE *
   HARMONY POSS
   POSS CASE *
```

The CASE morpheme can be the final morpheme identified during analysis; this is indicated by the presence of the asterisk following CASE. POSS and CASE follow the HARMONY morpheme; their forms, therefore, will be conditioned by that morpheme.

The tables in Fig. 6 (see next page) present a MORFOGEN analysis of these forms. Note that the HARMONY morpheme definition consists of a sequence of operations intended to establish the quality of the last stem vowel; these operations, or morphological expressions, are enclosed in parentheses and is assigned a unique name (ei and aI) (morphological expression operators are described in the next section). This holds for regular nouns; there is a class of nouns borrowed from Arabic or Persian which do not conform to vowel harmony; these would be marked in the lexicon, as involving an idiosyncratic HARMONY morpheme.

To ensure that the correct form of CASE follows the various forms of POSS, there are two POSS paradigms: the first lists the forms of CASE appropriate to the set of back unrounded possessive morphemes (referenced by the set name, BACK-UNROUNDED-POSS); the second is the paradigm of the front unrounded possessive morphemes (FRONT-UNROUNDED-POSS). A similar approach is implemented to ensure that the correct forms of CASE or POSS follow the HARMONY morpheme. Since no set of HARMONY morphemes has been declared, the actual HARMONY morpheme names are the paradigm names (see the tables for HARMONY CASE and HARMONY POSS).
MORPHOLOGICAL OPERATIONS

In the previous section we introduced MORFOGEN's morphological expression notation (cf. the zero morph, shown as \%null(), ). It is possible to assign these operations a unique name, by which they can be referenced when building paradigm tables for linked morphemes. This allows the abstraction of the actual operations from the analysis of the structure of lexical forms in the language, and at the same time allows the definition of operations having as their domain the entire word form, as will be shown.

Some operators take a string constant or variable as an argument; others are used without arguments. A variable can contain the character C or V optionally followed by regular expression operators (for example, "CVC", "C*VC*", or "C+VC+"). Currently, C and V are predefined symbols.

- "string" string is removed from the end of the word
+ "string" string is added to the end of the word
-1 "string" string is removed from the front of the word
+l "string" string is added to the front of the word
: "string" makes sure string matches end of the word (the word remains unchanged)
:1 "string" makes sure string matches beginning of the word (the word remains unchanged)
+ & & duplicates the final character of the word
- & & if the last two characters of a word are the name, it removes the latter of the two
- - "Z" where Z is a sequence of the following:
C 1 consonant
C* 0 or more consonants
C+ 1 or more consonants
V 1 vowel
V* 0 or more vowels
V+ 1 or more vowels
? any character
vowels and consonants are any from the regular DOS ASCII extended set; the given pattern is matched at the end of the word, and the matching string is removed and placed in a save buffer
adds the string in the save buffer to the end of the word
where Y is a sequence of characters sprinkled with periods for discontinuous infixation. For example, if Y is "me.e.h" and the word is "ktb", the new word will be "mektebeh". Filling starts from the right side of the word.
where Y is a sequence of characters sprinkled with periods for discontinuous deletion. For example, if Y is "me.e.h" and the word is "mektebeh", the new word will be "ktb". Y should match the end of the word.

The following examples will illustrate the use of MORFOGEN operators. The operations are executed from left to right.
a. German past participles

The analysis of German past participle forms may involve both a prefix, normally 'ge', and a suffix, normally 't'. The dictionary form of German verbs is typically obtained by adding the string 'en' to the form obtained after removing these affixes. For example, the past participle of 'leben' "to live" is 'gelebt', analyzed as 'ge+leb+t'. Fig. 7, part of the LEBEN paradigm, accounts for these facts:

```
< VERB PARTICIPLE LEBEN *
  leb en
  past-pple leb %(-! "ge" - "tt")
>
```

Fig. 7: Past participle form in the LEBEN paradigm

The operation shown will remove the string "ge" from the front of the word, and will remove the character "t" from the end of the word. Once the stem has been identified, the ending "en" will be added to it to generate the base form, which will then be looked up in the dictionary. This operation is not given a name, since this morpheme will not be used as the paradigm name in another table. Note, however, that it is the entire operation that defines the discontinuous PARTICIPLE morpheme.

b. Umlautung of German plural nouns

German nouns may umlaut the stem-final vowel nucleus when the plural ending 'e' is added. Thus, the plural of 'Baum' "tree" is 'Bäume'. The rule for the formation of the nominative plural of such nouns is show in Fig. 8:

```
< NOUN NUMBER-CASE BAUM PLURAL *
  Baum
  NOM Baum %(- "e" -- "C*" - "äu" + "au" ++)
>
```

Fig. 8: Umlautung in German nouns

This operation will remove the final 'e', remove (and store in the save buffer) any sequence of consonants, replace the sequence "äu" with "au" (if a separate character is used for the umlaut, this rule can simply remove it), and finally restore the string stored in the save buffer.

c. Arabic past participles

Arabic stems are three-character sequences (described as 'triliteral roots'); morphemes are discontinuous, consisting of characters interleaved among the three root characters. Thus, the definite masculine nominative singular form of the past participle of 'kataba' (the dictionary form of 'ktb'), "to read", is 'maktuwbu' (the root characters are boldfaced). Fig. 9 presents an analysis of this form using the % operators, which removes the interleaved characters, followed by %+, which introduces the characters required to generate the dictionary form; the morpheme is identified as DMSNPP (Definite Masculine Nominative Singular Past Participle):

```
< VERB PARTICIPLE REGULAR *
  kataba
  DMSNPP kataba %( %- "ma.uw..u" %+ ".a.a.a")
>
```

Fig. 9: Analysis of Arabic part participles

d. Degemination in English

This final example from English removes an 'ing' ending and removes the second of a sequence of two identical characters to the left of that ending:

```
< VERB V-INFL CUT *
  cut
  PROGR cut %( - "ing" - & &)
>
```

Fig. 10: Degemination in English

OPTIMIZING PARADIGM DESCRIPTIONS

MORFOGEN paradigms can be grouped in different levels; earlier levels are tried first, higher level rules being tried only if earlier level analyses failed. Levels are designated by integers preceded by the pound sign (e.g. #1, #4 etc.) This allows less frequent paradigms, or alternate forms for the same morpheme in a paradigm, to be assigned what amounts to a lower processing priority. Thus, the commonly used past tense form of 'burn', 'burned', can be described in level 1, while the more infrequent form 'burnt' can be assigned to a later level.

Fig. 11 shows the paradigms for verbs like 'walk' and for the verb 'burn' defined on level 1; the less frequent form 'burnt' is analyzed on level 2:
(ROOT-CATEGORIES (VERB))
(AFFIX-CATEGORIES (V-INFL))

# 1
< VERB V-INFL WALK *
walk
PROGR walk ing
3RD-SG walk s
PAST walk ed
PPLE walk ed
>

< VERB V-INFL BURN *
burn
PROGR burn ing
3RD-SG burn s
PAST burn ed
PPLE burn ed
>

# 2
< VERB V-INFL BURN *
burn
PAST burn t
PPLE burn t
>

Fig. 11: Specifying a paradigm on different levels

It should be noted here that the compiler takes cross-paradigmatic identity of forms into account: the compiled rules will not redundantly contain an ‘ed’ PAST for BURN and an ‘ed’ PAST for ‘walk’.

THE ROLE OF THE LEXICON

In addition to identifying putative base forms for a given inflected form, the analyzer will also look up these base forms in a dictionary. Analyses can then be rejected or accepted, if the lexicon includes inflectional information; additionally, morphemes can be entered in the lexicon and the information in those entries can be useful in determining the syntactic and semantic content of the form analyzed.

For instance, if the word ‘sing’ is marked as belonging to the inflectional class SING, the form ‘singed’ will not be analyzed as ‘sing+PAST’ but rather as ‘singe+PAST’. Similarly, the lexicon may contain two entries for ‘ring’, one meaning ‘to surround with a ring’ and marked as selecting the paradigm WALK, the other meaning ‘to produce a ringing sound’ and marked with the paradigm name SING. Note that, while a form like ‘rung’ will be analyzed as the latter, a form like ‘ringing’ or ‘rings’ will produce two analyses.

Specifying paradigms of derivational affixes is identical, in principle, to specifying paradigms of inflectional affixes. These paradigms can be assigned unique names, which can be used to mark words in the lexicon, alongside inflectional paradigms. In this manner, it is possible to constrain derivational analyses, ensuring that ‘belief’ will be the correct deverbal noun for ‘believe’, while ‘retrieval’ will be the correct form for the deverbal noun derived from ‘retrieve’, and rejecting forms like ‘believal’ and ‘retrief’.

All morphemes, including derivational morphemes, are assigned names; the analyzer returns these names along with the base form in the event of a successful analysis. The lexicon may contain entries for these morphemes. Each lexical entry for a morpheme can be represented as a feature structure, which can then be used by a unification-based parser to identify the morphosyntactic content of the inflected form (Antworth 91, Dalrymple 87). Derivational morpheme categories also may be defined in the lexicon, and may contribute their own feature structures to the representation of the inflected form. Thus, the analysis of a form like ‘antidisestablishmentarianists’ may identify the following morphemes:

antidisestablishmentarianists -->
establish + DIS + ANTI + MENT + ARIAN + IST + PLUR

The parser will then determine that this form is a noun derived from the verb ‘establish’, by virtue of the information in the morpheme lexicon. The role of the analyzer is limited to specifying allowable sequences and forms of morphemes; the actual content of the morphemes matched is properly represented in the lexicon and can be used to constrain derivations.
MORPHOGEN: MORPHOLOGY GRAMMAR BUILDER

PLATFORMS AND APPLICATIONS

MORPHOGEN is available on a variety of platforms (SUN, PC, 386-base UNIX/XENIX, and OS/2). The compact size of its compiled rules (50-60K) and its analysis routines allow its use in memory-resident mode even in systems with limited resources.

To date, MORPHOGEN’s analyzer has been used to interface with several natural language processing systems, including a retrieval system, several machine translation systems, an on-line dictionary lookup system, and a proofreader (currently under development). Analyzers have been written for English, Spanish, French, Japanese, Korean, Turkish and Arabic, with several other languages under development. Since MORPHOGEN’s analysis routines are also available in source code form, they can be integrated in other applications, allowing access to on-line dictionaries using any lexical definition scheme.

Describing the possible applications of two-level analyzers, [Koskenniemi 1984] made the following comments:

Systems dealing with many languages, such as machine translation systems, could benefit from the uniform language-independent formalism [of two-level models]. The accuracy of information retrieval systems can be enhanced by using [this] model for discarding hits which are not true inflected forms of the search key. The algorithm could be also used for detecting spelling errors. (p. 181)

We are happy to report that MORPHOGEN has to date proved to be at least as versatile as the two-level model, both in terms of its descriptive power and in terms of its range of applications.

CONCLUSION

The functionality and linguistic motivation of MORPHOGEN, a finite state morphological rule compiler and morphological analysis tool, was presented. Examples were adduced supporting our claim that MORPHOGEN allows the expression of linguistically sound descriptions for a variety of languages, using a hybrid of the morpheme/allo-morph approach and the Word and Paradigm model of morphological description.

MORPHOGEN’s compiler and analyzer are a standalone version of the morphological analysis routines of the ECS NLP/MT Toolkit. The morphological processor in that system performs both analysis and generation; we are currently working on including a generation capability in the standalone version.

Finally, our work developing analyzers for additional languages continues to suggest extensions to the list of MORPHOGEN’s operators.

ACKNOWLEDGMENTS

We would like to thank our colleagues Jay Kim and One-Soon Her for valuable comments and suggestions during the development of MORPHOGEN and during the writing of this paper; we would also like to thank Magnar Brekke for comments that helped us clarify the intent and coverage of MORPHOGEN as well as the purview this paper.

REFERENCES

