A Morphological Parser for Linguistic Exploration

David Weber
Summer Institute of Linguistics

1. INTRODUCTION

This paper describes AMPLE, a morphological parser (i.e., a program that parses words into morphemes). AMPLE grew out of work in computer assisted dialect adaptation, as described in section 1. It contains no language-specific code, being controlled entirely through external, user-written files, the notations of which were designed for linguists. AMPLE's constructs are linguistic: "allomorph", "morpheme", "conditioning environment", "co-occurrence constraint", etc.

AMPLE's fundamental algorithm is (i) to discover all possible decompositions of a word into allomorphs, and (ii) to eliminate those which fail any conditions, constraints or tests imposed by the user.

This match-and-filter algorithm allows a highly modular approach to morphological parsing. Strong rejection of incorrect analyses is achieved by the combined effect of diverse filters, each expressed simply in a notation appropriate to the phenomena.

AMPLE is a good tool for exploring morphology because of the flexibility resulting from this modularity. And it is usable by computationally naive linguists because its notations are linguistic rather than computational.

2. COMPUTER ASSISTED DIALECT ADAPTATION

Computer assisted dialect adaptation (CADA) attempts to exploit the systematic relationships between closely-related languages to produce drafts of text in target languages from source languages texts. (Initial explorations are described in Weber and Mann, 1979.) CADA works over non-trivial degrees of language difference because, between closely-related languages, most of the differences are systematic. These result from the generalization of regular diachronic changes, thus impacting the language heavily. By contrast, irregular or idiosyncratic changes cannot be generalized, so tend to have a limited impact. So between closely related languages, systematic differences predominate.

Differences are systematic only relative to some analysis. For example, between one dialect of Quechua and another, the character string *ra* might correspond to *ra*, *ri*, *ru* or *rqu*, but the context in which each is appropriate cannot be determined simply by inspecting adjacent character strings (in the source dialect text). However, if one can determine the identity of the morpheme in which *ra* occurs, the differences become systematic: when it is the past tense suffix, then it corresponds to *rqa*; when it is the punctual, it corresponds to *ri* or *ra*, depending on morphological context; when it is the directional 'out', it corresponds to *rqu* or *rqa*, and so forth.

Experience in various language families [Quechua, Tucanoan, Cakchiquel (Mayan), Campa (Arawakan), and the Philippine type] has shown that, for language families with rich morphologies, parsing words into morphemes makes most differences systematic, thereby providing a sufficient analytic base on which to do adaptation.

CADA's analytic engine began as a Quechua-specific morphological parser written in INTERLISP (Weber and Mann, 1979). This parser was re-implemented in C for small systems (Kasper and Weber, 1986a,b). This implementation was subsequently adapted to the Tucanoan language family of Colombia (Reed 1986, 1987), to Campa languages (Arawakan of Peru), and to Philippine languages. Guided by these extensions, a general morphological parser has been developed, called AMPLE (Weber, Black and McConnel, 1988).

AMPLE fits into word-by-word adaptation as indicated in Table 1:

```
                            +-----------------------------+ S
                            | +-----------+               | T
      word analyses-->      | | TRANSFER  |-->modified    | A
               |            | +-----------+ word analyses | M
  A +--------------+        +-------------+       |       | P
  M | +----------+ |                      | +----------+  |
  P | |          | |                      | |          |  |
  L | | ANALYSIS | |                      | | SYNTHESIS|  |
  E | |          | |                      | |          |  |
    | +----------+ |                      | +----------+  |
    |  normalized  |                      |  normalized   |
    |    words     |                      |    words      |
    | +----------+ |                      | +----------+  |
    | |  TEXTIN  | |                      | | TEXTOUT  |  |
    | +----------+ |                      | +----------+  |
    +--------------+                      +---------------+
           |                                      |
    source dialect text                  target dialect text
```

Table 1: THE MAJOR MODULES OF WORD-LEVEL CADA

The following illustrates how each module of Table 1 contributes to adapting from Pachitea Quechua *Aywarkaykargan* 'they were going' to the corresponding Huanca Quechua form, *Liyalkala*:

```
                 Pachitea: Aywarkaykargan
     TEXTIN                |          |
                           aywarkaykarqan
     ANALYSIS
                  aywa-   -rka      -yka     -rqa   -n
                 *aywa-   -PLIMPF   -IMPF    -PST   -3
                  |          \     /          |      |
                  |           \   /           |      |
     TRANSFER     |             X             |      ø
                  |           /   \           |
                  |          /     \          |
                 *ri-     -IMPF     -PLIMPF  -PST
     SYNTHESIS    |         |          |       |
                  li      -ya       -lka     -la

                           liyalkala
     TEXTOUT               |
                 Huanca: Liyalkala
```

In addition to serving as the analytic base for adaptation, AMPLE has been used to automate the glossing of texts (see, e.g., Weber 1987a), to detect spelling errors, and perhaps most significantly, to advance users' understanding of the morphology of various languages.

3. GENERAL AMPLE DESCRIPTION

Various external factors have shaped AMPLE: its constructs, mechanisms and notations must be familiar to linguists; its data files should be useful for other computational and non-computational purposes; it must run effectively on personal computers with small memories; and crucially, it must be able to cope with very diverse phenomena without unduly compromising linguistic integrity.

AMPLE takes text as input. It identifies words and normalizes them according to user-specified rules (e.g., change *b* to *p* before *m*). This allows the internal representation to differ from the external orthography (which might even be a phonetic representation). Each word is subjected to a depth-first, all paths analysis. The text is output as a database--one record per word--with fields for the (possibly ambiguous) analysis, punctuation, white space, format marking, and capitalization information.

AMPLE has various "biases." It is based on the assumption that morphemes exist. It applies directly to concatenative morphology; non-concatenative phenomena usually have to be coerced into concatenative solutions. For example, *took* could be analyzed as *take+PAST* (as suggested by Block 1947). To apply AMPLE to fusional languages generally requires large numbers of fused combinations constrained by declension or conjugation class. Finally, AMPLE takes an item/arrangement rather than an item/process approach (Hockett 1954). There are no "underlying forms" from which surface forms are derived.

AMPLE has main modules: SETUP, TEXTIN and ANALYSIS. SETUP reads files containing information about the language, creating internal structures for TEXTIN and ANALYSIS. Most significantly, SETUP reads one or more dictionaries, creates a trie structure based on allomorphs (character strings) for accessing the information about that allomorph and the morpheme it represents.

TEXTIN identifies the words of the text, putting to one side white space, capitalization information, format markup, and punctuation. User-specified orthographic changes are applied, allowing the internal working representation to differ from the practical orthography of the text.

Analysis parses by (i) discovering all possible sequences of matching allomorphs and (ii) filtering these with the tests that the user writes in various linguistically-oriented constraint languages (as described below). This proceeds bottom-up, left-to-right and exhaustively, i.e., all possible combinations of matching morphemes are discovered, and all which pass the tests are returned in the output. Matching and filtering are integrated so as to abandon false paths as early as possible.

There are two types of test. *Successor tests* apply when a matching allomorph is considered as the next possible morpheme of an analysis. *Final tests*, generally incorporating non-local dependencies, are deferred until an entire decomposition is discovered, one which passes all successor tests.

More specifically, as processing proceeds, a partial analysis is maintained. Whenever a matching allomorph is discovered, successor tests are applied between the partial analysis (usually its last morpheme) and the morpheme under consideration as a successor (for which some allomorph has been matched). For example, in analyzing *rikaykaamaran* 'he was watching me', the following stage would be reached:

```
                      see   IMPFV
                       |      |
   PARTIAL ANALYSIS: rika-  -yka:
 POSSIBLE SUCCESSOR:               -ma 1OBJ
   REMAINING STRING: maran
```

One of the successor tests, to take an example, insures that vocalic length (represented here as a colon) is not followed by syllable-closing suffix (since long vowels cannot occur in a closed syllable).

Successor tests have the advantage of eliminating false paths before they consume more computation, but they can not appeal to following morphemes, since these have not yet been identified. But final tests apply constraints to an entire analysis, so can express forward-referring constraints. For example, a final test might say that a morphophonemically affected unit must be followed (not necessarily adjacently) by a trigger for the process. Also, final tests can impose well-formedness constraints expressed on a particular morpheme; e.g. it might constrain the category of the final morpheme.

4. PHENOMENA

AMPLE can handle a wide variety of phenomena. Units may be prefixes, roots or suffixes, realized, null. or the reduplication of an adjacent segment Morphemes may have multiple allomorphs. AMPLE can handle the reduplication of adjacent segments (although the mechanism may be clumsy in some cases, as discussed below). Infixation is handled, even when obscured by prior or subsequent affixation or reduplication. The compounding of roots is handled (but nothing has been done to treat the compounding of morphologically-complex words).

4.1. Types of unit

AMPLE can deal with roots, suffixes and prefixes (of course!). More interestingly, it can deal with infixes, such as those of Philippine languages, for which an infix may be within a root or within a prefix, and where reduplication may apply after infixation. AMPLE allows compound roots, possibly constrained by the categories of those roots.

AMPLE allows null allomorphs. The occurrence of nulls must be strongly constrained, since they are not constrained by the characters of the word being analyzed. For example, in Napo Quichua, the agentive nominalizer has no phonological realization, due to its lenition and ultimate loss. But there is a strong constraint on its occurrence: it must be at a boundary where an uninflected verb is either word final or followed by suffixes typical of nouns. When adapting to Pastaza Quichua, where the agentive is /h/, it is thus possible to insert /h/ in the appropriate places with considerable accuracy. (For example, *rita* (= ri- 'go' -0 'agentive' -ta 'accusative', meaning 'to the one who goes') can become *ri-j-ta*.

4.2. Phonologically conditioned allomorphy

The occurrence of each allomorph in an analysis may be constrained by its phonological or morphemic environment, either locally or at a distance.

4.2.1. Issues of representation

The practical orthography of the text being analyzed may not be the best representation for doing analysis. (For example, in analyzing Spanish, it might be desirable to eliminate the orthographic alternation between *z* and *c* (cf. *raiz*, *raices*). Likewise, for Latin one might wish to convert *x* into *ks*, so that a morpheme boundary could be posited between the *k* and the *s* (cf. *rex* = /reks/, regis). Orthographic changes such as these can be made by the TEXTIN module.

4.2.2. Conditions on allomorphs

Allomorphs may be restricted by phonological (character string) environment. For example, the following says that *m* may only occur followed by *p*. (\a is the field code for "allomorph".)

```
    \a m / __p
```

Classes of phonological segments can be defined, and then used in constraining environments. For example, the following defines the class of labials and states that *m* must precede one of them:

```
\scl +labial p b f v
\a m / _[+labial]
```

4.2.3. Multiple allomorphs

Any morpheme may have multiple allomorphs. For example, the second person possessive in most Quechua languages have three allomorphs, constrained as follows (where ~[V] _ indicates "not following a vowel):

```
\a niki / ~[V] _ | hatunniki 'your big one'
\a ki   /    i _ | wasiki    'your house'
\a yki  /  [V] _ | umayki    'your head'
```

Reduplication is handled as a special case of multiple allomorphs, where each possibility is enumerated along with the environment in which it could occur (so, e.g., *pa* before *pa...*, *pe* before *pe*, etc. If the reduplicated from is always a precise substring of what precedes or follows, it is possible to state this as a general constraint rather than with each allomorph.

4.3. Morphophonemics

Phenomena involving both altered form (phonology) and morpheme identity present no special challenge because both the character string being analyzed and the posited morphemes are available.

4.3.1. Morpheme environment constraints on allomorphs

It is possible to restrict the occurrence of an allomorph by the identity of a morpheme; e.g., the following says that *an* must be directly followed by the morpheme identified as PQR:

```
\a an +/ _PQR
```

4.3.2. Properties and tests

It is possible to assign properties to allomorphs and morphemes and to use these in a very general constraint language. For example, suppose inherently applicative verbs may never co-occur with the applicative suffix APPL; this can be incorporated by assigning the property "applicative" to applicative verbs and imposing the following test:

```
IF   (current property is applicative)
THEN (FOR_ALL_RIGHT
      NOT (RIGHT morphname is APPL))
```

4.4. Morphotactics

AMPLE has good mechanisms for imposing morphotactic constraints There are three main types: categorial, ordering, and morpheme co-occurrence constraints.

4.4.1. Categorial constraints

Roots are assigned one or more categories, and affixes are assigned one or more category pairs. The left part of a category pair is called the "fromcategory" and corresponds roughly to the affix's "subcategorization frame." The right part is called the "tocategory" and corresponds roughly to its "category".)

In terms of these categories, tests can be imposed which "structure" the verb. To illustrate, consider a language with derivational suffixes (causative, applicative, passive, etc.) and inflectional prefixes. What inflection is permitted and/or required depends on the category after derivation, and "prior" inflection. Likewise, the derivational possibilities depend on the category of the root and any "prior" derivation. Thus, the constraints must propagate first progressively from the root through the suffixes and then regressively through the prefixes to the beginning of the word:

```
          S
         /\
        /  U
       /  /\
      /  /  Z
     /  /  /\
    /  /  X  \
   /  /  /\   \
  /  /  /  \   \
R/S T/U  V  W/X Y/Z
pfx pfx root sfx sfx
```

This can be achieved by four tests: (i) for suffixes (whereby V=W and X=Y above):

    left tocategory is current fromcategory

(ii) for prefixes (whereby U=R above):

    current tocategory is left fromcategory

(iii) to identify the category after derivation with that of the closest prefix (Z=T above):

```
IF (current type is prefix AND right type is root)
THEN (current fromcategory is FINAL tocategory)
```

(iv) to ensure that the category of the whole word (S above) is an acceptable terminal category, we can declare a class of such categories (called "finalcategories") and state:

    INITIAL tocategory is member finalcategories

Thus, although AMPLE processes from left to right, it is possible to model the percolation of features from a root through the layers of affixation, to the final resulting category of the word.

4.4.2. Ordering

The use of category along the lines described in the previous section may strongly restrict the order in which affixes occur. However, further ordering constraints may need to be imposed. This can be done by giving each affix a number (not necessarily unique) and imposing a successor test like the following:

```
left orderclass < current orderclass
```

This says that every morpheme's number must be greater than of the preceding morpheme, so insists that the orderclass strictly increase. If "<=" were used instead of "<", the order would be non-decreasing.

The test could also be modified to tolerate morphemes that are not constrained by order, such as Quechua *-lla* 'just'. To do so, we assign *-lla* orderclass 0, and then the following successor test passes it:

```
     (current orderclass = 0)
OR   (left orderclass <= current orderclass)
```

To make ordering constraints apply over one or more "floating" affixes, we give the following final test:

```
IF (    (current orderclass = 0)
    AND (FOR_SOME_LEFT  (LEFT  orderclass ~= 0))
    AND (FOR_SOME_RIGHT (RIGHT orderclass ~= 0)))
THEN (LEFT orderclass <= RIGHT orderclass)
```

4.4.3. Morpheme co-occurrence constraints

AMPLE has a simple but effective constraint language for imposing conditions on the co-occurrence of morphemes. The following, for example, says that PLIMPF can only occur preceding IMPFV:

```
\mcc   PLIMPF  / _ IMPFV
```

The following says that the conditional morpheme CND must be preceded (not necessarily contiguously) by a first, second, or third verbal person suffix (respectively named 1, 2, and 3):

```
\mcc CND / 1 ..._ / 2 ..._ / 3 ..._
```

The first line of the following defines a class of morphemes DIR, and the second says that PLDIR must precede a directional, the reciprocal or the reflexive:

```
\mcl DIR   IN OUT UP DWN
\mcc PLDIR / [DIR] _ / RECIP _ / REF _
```

5. AMPLE AS A TOOL FOR LINGUISTIC EXPLORATION

AMPLE has some features that enhance its usefulness as an exploratory tool:

1. It returns the original word (the \a field), that word's decomposition (\d), and the analysis (\a); for example, the following would be returned for *rirkansapanashi* 'they now went (it is reported)':

```
\a < V1 go > PST 3 PLUR NOW REPORT
\d ri-rka-n-sapa-na-shi
\w rirkansapanashi
```

2. AMPLE reports all analytic failures, indicating how far into the word it was able to proceed and whether or not it matched a root. This often provides a sufficient clue to why the word failed to be analyzed. For example, the following report (for Quechua) makes it clear that (i) the root *fes* (*hwes* after orthography changes) is not available as a root, and (ii) there is an incompatibility between the suffixes *-ri* and *-ma:*:

```
        Root Failure: hwesqa   [ | fesqa ]
    Analysis Failure: roqorimaachun  [ roqori | ma:chun ]
```

3. AMPLE reports on the effectiveness of each test: for both the user-defined and built-in tests, it reports how many times each test was applied (in the order of application) and how many analyses were filtered out by the test:

```
   CATEGORY_ST called 10936 times, failed 7436.
      ORDER_ST called  3500 times, failed 392.
FORESHORTEN_ST called  3108 times, failed 36.
    MLOWERS_ST called  3072 times, failed 2.
```

4. The user can control which tests are applied and the order of their application. This makes it possible to see the effectiveness of each, and their joint effect.

5. Ambiguity levels are reported as follows:

```
  2 words with 0 analyses.
620 words with 1 analysis.
 73 words with 2 analyses.
  2 words with 3 analyses.
  3 words with 4 analyses.
```

6. It is possible to trace AMPLE's parsing activity. For example, the following is the first part of the trace for the Quechua word *nimaran*:

```
Parsing nimaran
root: ni, *ni V2
sfx: ma, 10, V2/V1, order: 70, ullong Mlowers, fshrtnd
  sfx: ra, PST, V1/V1, order: 80, foreshortens
    sfx: n, 3P, N0/N0, order: 140   / [V]__
            Suffix test CATEGORY_ST failed.
    sfx: n, 3P, R1/R0, order: 140   / [V]__
            Suffix test CATEGORY_ST failed.
    sfx: n, 3P, N1/N0, order: 140   / [V]__
            Suffix test CATEGORY_ST failed.
    sfx: n, 3, V1/V0, order: 120, foreshortens
            No more suffixes found.
            End of word found; checking final tests
                 Analysis string: < V2 *ni > 10 PST 3
                 Decomposition:   ni-ma-ra-n
```

After achieving this analysis, AMPLE continues considering other possibilities.

A future version of AMPLE will allow selectivity in tracing, more information in the analysis (e.g., the category pairs used in an analysis), and quantifying the contribution of specific morphemes, tests, etc. to analysis.

6. CONCLUDING REFLECTIONS

AMPLE's match-and-filter algorithm permits a highly modular approach to morphological parsing. Strong rejection of incorrect analyses can be achieved by the combined effect of diverse filters, each of which may be quite simple. Direct reporting of these linguistic constraints is possible because they are not compiled into some inaccessible form. And this algorithm has proven to be reasonably efficient.

Our success with the match-and-filter algorithm suggests that morphology has a modular organization. That is, the organization of morphology may resemble the Chomskian approach to syntax, where diverse principles or theories, here expressed as filters, jointly but modularly define acceptability.

Each filter is expressed simply in a notation appropriate to the phenomena and familiar to the users, in this case linguists. This makes it quite straight forward for linguists to set up a morphological parser for a language. Experience has repeatedly shown that doing so leads the user to new insights into the morphology. Because there are various constraint languages and mechanisms, AMPLE can be used to model various conceptions of the morphology, and to quickly test these against large amounts of data.

The modularity afforded by the match-and-filter approach also makes AMPLE very extensible: as other constraint languages are discovered (and notations developed) they can be integrated into AMPLE. For example, we are considering an alternative (or complement) to the category system that would allow categories to be defined as sets of features, incorporating percolation, redundancy rules and feature addition rules; see in Weber 1987b.

We expect AMPLE to be useful in conjunction with various syntactic parsers. In one experiment, a unification-based parser (adapted from an early version of PATR-II) parses sentences (or sentence fragments) using AMPLE output. The morpheme dictionaries, are read once by AMPLE for the morphological information and again by the syntactic parser for the syntactic parser.

We hope that in the next few years AMPLE will be applied to a much wider range of languages.

current address: David Weber
6004 Stanton Ave. A-15
Pittsburgh, PA 15206

REFERENCES

Block, Bernard. 1947. "English verb inflection." **Language**. 23:399-418.

Hockett, Charles. 1954. "Two models of grammatical description." **Word**. 10:210-231.

Kasper, Robert and D. Weber. 1986a (written in 1982). **User's Reference Manual for the C Quechua Adaptation Program**. Occasional Publications in Academic Computing No. 8. Dallas: Summer Institute of Linguistics.

__________ and __________. 1986b (written in 1982). **Programmers Reference Manual for the C Quechua Adaptation Program**. Occasional Publications in Academic Computing No. 9. Dallas: Summer Institute of Linguistics.

Reed, Robert. 1986. **Computer Aided Dialect Adaptation: The Tucanoan Experiment**. PhD dissertation in linguistics. University of Texas at Arlington.

__________. 1987. **The Implementation of a System for Computer Aided Dialect Adaptation**. MA thesis in computer science. University of Texas at Arlington.

Weber, David (editor). 1987a. **Juan del Oso**. Serie Linguistica Peruana No. 26. Yarinacocha: Instituto Linguistico de Verano.

__________. 1987b. "Sobre la morfologia quechua." **Estudios Quechua**. Serie Linguistica Peruana No. 27. Yarinacocha: Instituto Linguistico de Verano.

__________ and William Mann. 1981. "Prospects for Computer Assisted Dialect Adaptation." **AJCL**. 7:165-177.

__________, H.A. Black and S.R. McConnel. 1988. **AMPLE: A Tool for Exploring Morphology**. Occasional Publications in Academic Computing No. 12. Dallas: Summer Institute of Linguistics. (available from the ILC Academic Book Center; 7500 W. Camp Wisdom Rd.; Dallas TX 75236).