David W. Packard

COMPUTER-ASSISTED MORPHOLOGICAL ANALYSIS OF ANCIENT GREEK

INTRODUCTION[1]

This system for automated morphological analysis of ancient Greek had its origin in a practical need rather than a theoretical concern for natural language parsing. Our immediate goal was to develop a new textbook and curriculum for teaching ancient Greek to American university students. Most traditional methods assume that a student is willing to spend at least one year in the study of grammar before reading any significant quantity of literature. Our conviction is that students can begin reading literature very early in the first year if the initial grammatical instruction is focused on those features of the language which actually occur in the texts first read. To test this theory, we have used the computer to help produce a complete lexical and grammatical analysis of 40,000 words of ancient Greek selected from texts which students might wish to read in their first year. We have concentrated our attention initially on morphological analysis since the complexity of Greek morphology is the major obstacle to learning the language. We have prepared statistical summaries of the morphology of each text, as well as complete concordances organized both according to dictionary lemma and morphological category. Although our first goal is to collect information for a textbook, it is obvious that an automated system for morphological analysis will have uses far beyond the teaching of elementary Greek.

THE METHODS OF ANALYSIS

Our system is based on a combination of computer analysis and subsequent editorial verification. The program is able to identify most

[1] I wish to thank the University of California for sponsoring this work under its program *Innovative Projects in University Instruction.*

Greek words automatically, but we always examine carefully the resulting analysis and make provision for correcting and supplementing it manually wherever necessary.

No prior editing of the text is required; the program will accept any Greek text which includes the normal diacritical signs (accents, breathing marks, and iotas subscript). The words are analyzed in the order in which they appear in the text without being sorted into alphabetical order. Each word is first examined to determine whether it occurs in a list of exceptional forms. If the word is found in this list, it is not subjected to further analysis. This list, which we call the *indeclinable list*, contains forms which are not inflected (prepositions, adverbs, particles, etc.) as well as forms whose inflection is highly irregular. The list currently contains about 800 entries. Roughly 50% of the words in a typical text are found in this list.

Words not in the indeclinable list must be analyzed according to the rules of Greek morphology. Since the analytical procedure has refinements peculiar to Greek, it cannot be understood without some knowledge of how Greek nouns and verbs are inflected.

Most inflection in Greek consists of adding an *ending* to a fixed *stem*. The present active indicative of the verb γράφω 'write' is conjugated: γράφ-ω, γράφ-εις, γράφ-ει, etc. The task of analyzing such forms is simply one of segmenting the word into a stem and an ending. The program first removes the final letter of the word and determines whether that letter appears in the table as an inflectional ending. If it does, the remainder of the word is tentatively assumed to be a stem, and a search is made for this stem in the dictionary. If the stem exists and is consistent with the ending, the program identifies this as one possible analysis. The program then continues by searching for longer endings. Each possible combination of stem and ending must be examined since the word may be ambiguous.

In some cases the original juncture between stem and ending is obscured by phonological changes. The verb ἀγαπάω 'love' was originally conjugated with the same endings as γράφω: ἀγαπά-ω, ἀγαπά-εις, ἀγαπά-ει, etc., but in the standard literary dialect of Athens (Attic) the adjacent vowels contract, giving the following forms: ἀγαπῶ, ἀγαπᾷς, ἀγαπᾷ, etc. We treat these contracted forms as a separate conjugation with a separate set of endings, even though this produces a false division between stem and ending: ἀγαπ-ῶ, ἀγαπ-ᾷς, ἀγαπ-ᾷ, etc. With the proper selection of such pseudo-stems, it is possible to break down nearly any inflected form into a stem and an ending. Especially troublesome are nouns of the third declension where the nominative singular and dative plural are subject to a great variety of sound changes. The nominative νύξ 'night', for example, and its dative plural νυξί must be placed in the indeclinable list since they cannot be reconstructed directly from the stem νυκτ-. Words like μάθημα 'lesson', however, are broken into a pseudo-stem μαθη- and the endings -μα, -ματος, -ματι, -ματα, -ματων, -μασι. This allows us to avoid the need to enter each nominative singular and dative plural into the exception list, as would have been necessary with the linguistically correct stem μαθηματ-. Such decisions of expediency do not affect the final analysis but only the construction of the tables used by the program.

A Greek verb can have six *principal parts*; these are the stems which form the basis for conjugation. The present stem of γράφω, for example, is γραφ-, the future γραψ-, the aorist γραψ-, the prefect γεγραφ-, the aorist passive γραφ(θ)-. Past tenses of the indicative (aorist, imperfect, pluperfect) *augment* these stems by prefixing an initial ε-, or if the stem begins with a vowel, by lengthening that vowel. The imperfect indicative built on the stem γραφ- is ἔ-γραφ-ον, ἔ-γραφ-ες, ἔ-γραφ-ε, etc. The aorist indicative built on the stem ἐλθ- is ἦλθ-ον, ἦλθ-ες, ἦλθ-ε, etc. In an earlier version of our program we included in the dictionary both augmented and unaugmented forms of each stem (γραφ- and ἐγραφ-, ἐλθ- and ἠλθ-). This was uneconomical since the augmented form is nearly always predictable. The dictionary now contains only unaugmented stems except for a few verbs like ἔχω and ὁράω which are augmented in special ways (εἶχον and ἑώρων). Reduplicated perfect stems, however, are entered in the dictionary.

Greek shows great freedom in forming compound verbs by the addition of prepositional prefixes. From the stem γραφ- is derived παρα-γραφ-ω 'write beside', κατα-γραφ-ω, 'write down', ὑπο-γραφ-ω 'write under', etc. It would be uneconomical to include παραγραφ-, καταγραφ-, and ὑπογραφ- in the dictionary since all are formed by the addition of common prefixes to the single verb stem γραφ-. A difficulty arises, however, from the fact that the prefixes are often assimilated phonetically to the following letter. The prefix συν- 'together' appears as συν- before vowels and dental consonants, as συμ- before labial consonants, as συγ- before guttural consonants, as συλ- before λ, and as συ(ς) before ς. The prefix μετα- appears as μετα- before consonants, μεθ- before vowels with aspiration and μετ- before vowels without aspiration. The program must recognize the assimilated forms of each prefix and must verify that the letter following the prefix could in fact have caused the suspected assimilation. In some cases a single verb is compounded with as many as three prefixes, each of which may appear in an assimilated form. The form συγκαθίστημι must be analyzed as συν + κατα + ἵστημι, συνεπανίστημι as συν + ἐπι + ἀνα + ἵστημι. Further complication is caused by the fact that verbal augments come before the stem but after the prefixes. The imperfect of συμ-βαίν-ω is συν-έ-βαιν-ον.

Thus, if the word cannot be analyzed directly into a stem and an ending, the program must attempt to remove prepositional prefixes from the beginning of the word. If a hypothetical prefix can be removed, the program proceeds to analyze the remainder of the word. If this analysis is successful the prefix is reunited with the word in the final analysis. In some cases the program makes more than one hypothetical division between prefix and stem. The verb ἀνα-λύω would generate three hypothetical divisions: ἀνα + ἀλύω, ἀνα + ἁλύω, and finally ἀνα + λύω.

If the word still cannot be analyzed, the program attempts to isolate a verbal augment at the beginning of the stem (but after any prepositional prefixes). Most imperfect and aorist indicative verbal forms are analyzed only at this point. The program often generates several hypothetical unaugmented stems. The imperfect indicative of ἄγω is ἦγον. In analyzing this form, the program would make two initial false attempts: augment + ἠγ-, augment + ἐγ-, before finding the correct analysis augment + ἀγ-. Both prefix and augment may be ambiguous. The form παρήτουν would produce eight hypothetical divisions:

παρα + ἠτ-,
παρα + ἡτ-,
παρα + augment + ἠτ-
παρα + augment + ἡτ-
παρα + augment + εἰτ-
παρα + augment + εἱτ-
παρα + augment + αἰτ-
παρα + augment + αἱτ-.

Final short vowels are often elided, especially in poetry. If the program finds an apostrophe at the end of a word, it hypothetically restores each short vowel in turn. Most elided forms can be reconstructed successfully by this method. Crasis, the merging of two words into one, is more difficult to recognize automatically. We simply enter the most common examples (e.g. τἀληθῆ, ἐγῷμαι, etc.) in the indeclinable list and leave the others for the editor.

A major problem faced by any automated analysis procedure is how to deal with ambiguity. The program often generates more than one potential analysis for a single form. It would be possible to print every alternative, but we have found it more satisfactory to print only the most likely analysis together with a warning of possible error. Mistakes can be corrected later by the editor. There seems to be little ambiguity in Greek about which stem should be assigned to each form. Examples of such ambiguity are more interesting as curiosities than obstructive to the analysis process. Most of them occur only because the program ignores accents. The following 'ambiguous' forms were found in the first 2000 words of Euripides' *Medea*: φυγη (φεύγω or φυγή?), φιλων (φίλος or φιλέω?), οἰκων (οἶκος or οἰκέω?), τεκνων (τέκνον or τεκνόω?), ἀρχη (ἄρχω or ἀρχή?), μοχθου (μόχθος or μοχθέω?), ἀγων (ἀγών or ἄγω?), αἰων (αἰών or αἶα?), φοβου (φόβος or φοβέω?), ἐξιασι (ἔξειμι or ἐξίημι?), καλων (καλός, κάλως, or καλέω?), κοινωνων (κοινωνός or κοινωνέω?), ἐξεδου (ἐκδίδωμι or ἐκδέω?), σωφρονων (σώφρων or σωφρονέω?), ἀκουσαι (ἀκούω or ἀκέομαι?), τεκνοις (τέκνον or τεκνόω?), πονων (πόνος or πονέω?), τυραννων (τύραννος or τυραννέω?), βοαν (βοή or βοάω?), τελευτα (τελευτή or τελευτάω?), ὁρκων (ὅρκος or ὁρκέω?). In nearly every case of ambiguity between a verbal and a nominal analysis, we have found that the nominal form is correct. The program accordingly always prefers the latter.

Much more common are ambiguities about whether an article or adjective is masculine or neuter, whether a neuter noun is nominative or accusative, or whether a verb is middle or passive. Here again, we allow the program to make a likely guess and mark the analysis as doubtful. The editor can correct any mistakes.

A certain amount of this second kind of ambiguity could undoubtedly be resolved rather easily by a primitive syntactical scan which limited itself to immediate constituents of the type article + adjective + noun or preposition + noun. The gender of the noun in the first case would often make clear the gender of the adjective and article.

DETAILS OF IMPLEMENTATION

In line with our pragmatic approach and our immediate need to have a large volume of text analyzed, we have designed our procedures to take advantage of the hardware available at the UCLA Campus Computing Network; but the program could be modified to operate on other systems without great difficulty.

Our first problem was the representation of the Greek alphabet with accents and diacritical signs on a keypunch. Here are the first few lines of the *Apology* as we keypunched them:

```
O(/TI ME|N U(MEI = S, W) = A)/NDRES *)AQHNAI = OI,
PEPO/NQATE U(PO| TW = N E)MW = N KATHGO/RWN,
OU)K OI) = DA: E)GW| D' OU) = N KAI| AU)TO|S U(P'
AU)TW = N O)LI/GOU E)MAUTOU = E)PELAQO/MHN,
OU(/TW PIQANW = S E)/LEGON.
```

This transliteration suffices for most purposes; we have a Greek font on an RCA Videocomp photocomposer available whenever we wish to print the Greek in a more conventional manner.

The program is written in Assembly language for an IBM 360/91 computer. In the current implementation all tables reside in core storage. With 300K bytes of memory, we can accomodate 4000 stems, 2000 endings, and 1000 indeclinables. This is the maximum memory allowed for high priority jobs at our computing center, but our machine will accept jobs requiring several million bytes on a low priority basis. Using more sophisticated data compression techniques, we could expand the table size to around 10,000 stems without exceeding 300K of storage. So far we have been able to work easily within the limitation of 4000 stems.

The dictionary lookup uses a simple binary search algorithm. The tables are sorted by the program at the beginning of the analysis. This allows extra items to be inserted temporarily for each text without making them permanent members of the dictionary stored on the disk. The morphological endings, however, are stored in a permanent tree structure. This not only conserves storage but also allows much faster searching. The tree, which currently contains about 2000 endings, is generated by a separate program. The analysis program reads it from the disk along with the lists of stems and indeclinables.

Ad hoc programming techniques are used for isolating the prefixes and augments. Greek shows many idiosyncrasies in this area, and it would be difficult to design an efficient table-driven algorithm for this purpose.

If this system is to be expanded to include dictionaries for a wider range of texts, it will eventually be necessary to exceed the current limitation on dictionary size. At our computing center the existing program could accomodate at least 25,000 stems, which is undoubtedly adequate for any text. If the system were to be used on a machine with limited memory, however, it would be desirable to keep the entire list of endings in memory at all times and to process the text several times with different subsets of the dictionary.

THE FIRST FORTY-THOUSAND WORDS ANALYZED

Rather than encoding a standard list of Greek verbal and nominal stems, we have decided to add entries to our dictionary only as they are needed. This approach seems satisfactory, and our tables now suffice for a wide range of Greek prose of the classical period. So far we have analyzed Plato's *Apology*, *Crito*, and *Lysis*, Xenophon's *Constitution of the Spartans*, Euripides' *Medea*, and selections from Thucydides and the New Testament. In the course of analyzing these texts, we have been constantly improving the dictionary and the program. We expect that far less time will be required in correcting the computer analysis in the future.

Our tables were developed entirely from the texts cited above. It is encouraging that these tables appear to include a very high proportion of the forms needed for analyzing a variety of additional texts, from the Hippocratic corpus to the fables of Aesop. A problem perhaps unique to Greek is the existence of several literary dialects with varying morphology. Within a single Attic drama, the dialogue is in Attic while the choral odes are in a form of Doric. Moreover, Herodotus and the Hippocratic authors write in Ionic, and Homer requires yet another set of endings and dictionaries. We are currently working on tables for use in analyzing Homer.

The program is able to analyze about 2000 words per second. The computer charges for analyzing the entire *Apology* are under one dollar. We have found that we require about one hour of manual editing to check 500 words of analyzed text, or about four hours of human time to check the work done by the machine in less than one second. The cost of this editing is about fifteen times greater than the cost of the initial computer analysis. For this reason we have taken pains to design a convenient system for introducing corrections into the analyzed text.

Our program has many points in common with the system used by Professor Delatte in Liège for automated analysis of Latin. So far as I know, the Liège program does not include mechanisms for identifying augments and assimilated prefixes, but these features could undoubtedly be added. Despite the conceptual similarities, there are significant differences between the two systems in terms of implementation. Our program, since it maintains all of its tables in core storage, requires nearly twenty times more memory than theirs and could not possibly be used on a small computer like their IBM 360/20. The Liège system, on the other hand, could be used on a large machine like ours, but it would generate a tremendous amount of superfluous input-output activity by searching the dictionaries on the disk. Each program takes advantage of the characteristics of the machine for which it was designed. Given a large machine, our approach is probably more economical, but the computer charges are a small part of the total expense involved in producing an accurate analysis.[2]

An entirely different approach is taken by Busa who generates all possible inflected forms of each stem and then collates his sorted text with this list. I can see no advantage in using this method on machines with large core capacity. Busa's approach, on the other hand, might be appropriate for machines with extensive magnetic tape capacity but limited core storage and no disks.

INITIAL RESULTS

As an example of our statistics, we can compare the morphology of Plato's *Apology* with the grammatical topics introduced in the first thirty lessons of a widely used elementary Greek textbook. The first lesson (Lesson 3) equips the student to recognize the endings of more than 1500 of the 8000 words in the text (the definite article, ἀγαθός, λόγος, γνώμη, and δῶρον). The next lesson introduces adjectives like ἄξιος with 250 examples, the relative pronoun with about 100, and various first declension nouns which account for an additional 150 forms. The first chapter on verbs brings 200 verbal forms within the student's grasp, and Chapter 8 increases this by 150 imperfect and

[2] The Liège program is described in great detail by JOSEPH DENOOZ in the *Revue* (1973, No. 1) of the *Organisation Internationale pour l'Etude des Langues Anciennes par Ordinateur.*

aorist indicatives. In the first twelve lessons, little is superfluous, though one might question the presentation of σοφώτερος and ἀμείνων in Lesson 9. Lessons 12 and 13, however, enable the student to recognize exactly five new words in the *Apology*. These chapters present the -μι verbs. These uncommon, confusing, irregular verbs are introduced far earlier than the middle voice and the participle, each of which occurs many times more often. Lesson 15 consists of τιμάω and φιλέω, which covers 150 forms. Lesson 16, however, adds only 9 forms, apart from the very common personal pronouns (about 500 examples). The active participle is postponed until Lesson 19, despite its great frequency (230 occurrences), and the extremely common middle voice is presented only in Lessons 25 through 29. A textbook designed to equip students with the morphology needed for reading Plato early would be organized far differently.

On the basis of these statistics, we have written (and are now using) a new textbook which introduces the middle voice (Lesson 4) and the participle (Lesson 6) very early but postpones -μι verbs and other less common forms.

APPENDIX

The purpose of this Appendix (written in 1977) is to describe briefly our progress since 1973. Many additional texts have been analyzed and corrected, including four books of Thucydides, the *Hellenica* of Xenophon, and the *Phaedrus*, *Gorgias* and *Laws* of Plato. The program has successfully been installed at half a dozen universities. The editing of the analysis is now peformed on consoles equipped with the full Greek alphabet including accents (on an *Ibycus System*).

There has also been a major revision in the internal organization of the dictionary used by the program. The new dictionary is created by a computer program from a dictionary in the old format. Those who use the analysis program do not need to be aware of the internal structure of the dictionary, but it may be of interest to specialists.

Each entry in the stem dictionary consists of four elements: the *stem*, the *type-code*, the *flag*, and the *lemma*. In the initial version of the program these fields were stored in a fixed format:

Stem	*Type*	*Flag*	*Lemma*
ἀνθρωπ	N2	M	ἄνθρωπος
βελτιστ	A2	S	ἀγαθός
βουλ	V1	M	βούλομαι
βουλ	N1	F	βουλή
πολι	N3I	F	πόλις
γραφ	V1		γράφω
γραψ	VF		γράφω
γραψ	VA		γράφω
γεγραφ	VX		γράφω
γεγρα	VP		γράφω
γραφ	VC		γράφω

This format is convenient for external use, but it is a very inefficient way to store the dictionary inside the computer. We must allow fourteen characters for a long word like πολυπραγμοσυνη, but we need only five for a short word like γραφω. The same objection holds for the type field which uses three characters to distinguish fewer than one hundred different codes, and the flag, which has only four possible values. In the case of the lemma field, not only does the fixed length format waste space for short lemmas, but the same lemma is often repeated with more than one stem. Many lemmas, moreover, can be reconstructed from the stem simply by adding an ending and an accent.

The Stems

In order to conserve space, the stems are divided into fifteen sub-dictionaries according to their length. All stems of length one are stored in alphabetical order in the first subset of the dictionary, followed by stems of length two, and so forth. This solves the problem of wasted space, but it requires a table showing the location of each sub-dictionary. In order to find a given stem in the dictionary, we use the *Stem Length Table* to find the location of the dictionary for stems of that length. Within the proper sublist the stem is found by a binary search.

The Type Code

Associated with every stem is a one-byte field specifying the type. Internally the types are coded as follows:

1xxx xxxx	Verbs	128 types
01xx xxxx	Nouns	64 types
001x xxxx	Adjectives	32 types
0001 xxxx	Pronouns	16 types
0000 xxxx	Others	16 types

The verb stems are further subdivided into the following classes:

10 000 xxx	present stems
11 000 xxx	augmented present stems
10 001 xxx	present stems
11 001 xxx	augmented present stems
10 010 xxx	aorist stems
11 010 xxx	augmented aorist stems
10 011 xxx	aorist passive stems
11 011 xxx	augmented aorist passive stems
10 100 xxx	perfect active stems
11 100 xxx	augmented perfect active stems
10 101 xxx	perfect middle stems
11 101 xxx	augmented perfect middle stems
10 110 xxx	future stems

The last three bits (printed above as xxx) provide eight types within each class. Two present classes accommodate sixteen types. The second bit from the left always indicates whether the stem is augmented. Since a future stem cannot be augmented, the type code 1111xxxx is illegal and reserved for special use internally by the program (for duplicate stems).

The Flags

Each stem has a flag with four possible values. The meaning of the flag depends on the type of the stem. For nouns the flag gives the gender. For verbs the flag indicates whether the stem is deponent. For adjectives the flag identifies stems that are by nature comparative or superlative. Internally the flag is coded as a two-bit binary number and is stored in the leftmost portion of the lemma field.

The Lemmas

Lemmas are divided into two basic classes, those that are *self-defining* and those that are not. A self-defining lemma can be reconstructed by adding one of the following endings to the stem:

α	ον	μα	αω
η	ων	ω	οω
ης	ις	ομαι	ς
ος	ευς	εω	

The suffix is coded as a four-bit binary number. Another four-bit number specifies which accent to use and where to place it. These two four-bit fields are packed into a single byte. A self-defining lemma is recognized by the fact that the first byte contains ff111111 (where ff are the two flag bits). The second byte contains the suffix and accent codes. Lemmas which are not self-defining are stored in the *Lemma String.* Each lemma appears only once in this string. The lemma field in the stem list is an index into the *Lemma Table,* which gives the length of the lemma and its origin within the Lemma String.

Duplicate Stems

It often happens that two or more stems are spelled alike. In such cases only one copy of the stem is placed in the dictionary itself. This stem, instead of having its type code and lemma code, has a pointer to the *Duplicate Stem List.* Duplicate stems are recognized by a type code of the form 1111*nnnn*, where the four-bit binary number *nnnn* serves as a counter showing how many entries for this stem exist in the duplicate stem list. The lemma field is an index into this list. Each entry in the duplicate stem list is a three-byte field consisting of a standard type code and lemma field.

Savings in Space

In the original fixed format the six stems of the verb γράφω required a total of 240 bytes in the dictionary (six entries of 40 bytes each). With the new format the space required is as follows:

The stem γραφ	7 bytes
The stem γραψ	7 bytes
The stem γεγραφ	9 bytes
The stem γεγρα	8 bytes
Duplicate list for γραφ	6 bytes
Duplicate list for γραψ	6 bytes
Lemma string for γράφω	6 bytes
Lemma Table for γράφω	4 bytes

The total space required is now 53 bytes, or about 20% of the previous value. For many nouns and adjectives the saving is even greater. The noun θεός now requires only five bytes (the lemma is self-defining). In a large dictionary the average entry requires about eleven bytes.