AN ENGLISH AFFIX ANALYZER WITH INTERMEDIATE DICTIONARY LOOK-UP

Jonathan Slocum

Working Paper
February 1981
LRC-81-1

ABSTRACT

A program for removing any of a set of "productive" English affixes is described. The program is claimed to be efficient in that it combines context with empirically-derived probabilistic information to determine the best (first) guess. The guess must be verified via dictionary look-up before it can be accepted; furthermore, the affix must be appropriate to some syntactic class(es) of the proposed root.

INTRODUCTION

This paper documents a production INTERLISP program for affix separation in English text. The program considers only what are called "productive" affixes -- those admitting algorithmic determination of the meaning of the original word given the root and affix(es), without recourse to special, "lexical" flags or markers. As an added bonus, it expands contractions (-N'T, -'RE, -'VE, -'LL, -'M, -'D, -'S) into their full forms, separates any appended punctuation, and converts the determiner `AN' to `A'. This program began as an implementation of the flowchart presented by Winograd (1972); in the process of testing and evaluation, several bugs were excised and the algorithm was made more efficient. Subsequently, its capabilities were significantly extended.

The analyzer handles a complete range of regular verb endings (-S, -ING, and -ED), noun endings (-S, -'S, and -S'), adjective and adverb endings (-ER, -EST, and -LY), and ordinal number endings (-ST, -ND, -RD, and -TH) -- some of which involve minor spelling changes (e.g., consonant doubling). In addition, it handles almost all of the -EN verb endings plus many irregular verb forms exhibiting internal vowel shifts, as well as several noun plural variants drawn from Latin or Greek (e.g., FORMULAE, QUANTA, THESES, SCHEMATA). If all other attempts fail to produce a root, the program checks for one of several negative prefixes (e.g., NON-, UN-); if found, the prefix is removed and the remainder is once again checked for a suffix. In all cases, if the proposed root scores a lexical "hit", a test is performed to insure that the suffix is in fact appropriate to at least one sense of the root: if the test fails, the interpretation is rejected and others may be tried. (This strategy also serves to disambiguate roots with multiple senses, based on syntactic class.) If all attempts fail to disclose a root, the analyzer calls the user-supplied function SPELLING-ERROR to take appropriate action.

The first function of the analyzer is always to seek a lexical entry for the complete word; but even if one exists, affix stripping is attempted due to potential ambiguities. The stripper returns the first successful "hit" -- no alternate stripping is performed once a lexical root appropriate to the affix(es) has been found; this is combined with any "unstripped" analysis to constitute the list of analyses returned by the top-level function LEX.LOOKUP. It is intended that the analyzer be used to return affixes (if any) plus word senses syntactically appropriate to the root and its suffix. In line with this practice, the user is responsible for providing [1] a lexicon of roots and certain "exceptional" variants (e.g., CHILDREN), along with [2] a definition for the function LEXENTRIES to take a word or proposed root as its first argument, a list of category restrictions as its second, and return a (possibly empty) list of senses (definitions, of whatever nature) as its value, and optionally, [3] definitions for the functions SPELLING-ERROR, LEXBEG, and LEXEND, for which defaults are provided.

For handling all regular affixes, the program reverses a fairly well-known set of rules governing consonant doubling, changing Y to I, etc. For handling irregular verb forms, certain rules have been empirically derived from "families" of vowel shifts and deletions in English verb morphology: this derivation resulted from a study of what is thought to be a complete list of irregular English verbs (barring compounds which inflect like the verb root, e.g., BEGET, FORBID, like GET, BID). Appendix I lists all the irregular verbs, some forms of which the analyzer can handle by rule. Appendix II lists all irregular verb forms that are known to be "exceptional." The analyzer should be capable of reducing any other verb forms to their roots, provided the user's lexicon includes all the roots (infinitive forms) of the verbs that will be encountered. The analyzer should also be capable of reducing all regular noun plurals to their singular forms (again, with the user's lexicon caveat); however, no comprehensive list of noun exceptions (e.g., CHILDREN) is included.

ROOT-SUFFIX AGREEMENT

Associated with each canonical suffix (ED, EN, ER, EST, GEN ['], ING, LY, NEG, -N, -S, TH) and each expanded contraction (AM, ARE, HAVE, IS, NOT, WILL, WOULD) is the attribute P.O.S; the value associated with the attribute is a list of categories indicating which parts of speech of the root may appear with the suffix or contraction. The user function LEXENTRIES will receive such a list as its second argument, and should return the subset of the sense meanings of a proposed root (the first argument) which can accept the suffix. If the subset is empty, then the proposed root is rejected, and another interpretation will be sought. That is, in any case where the P.O.S attribute does not agree with the syntactic category of one or more senses of the proposed root, then either [1] the entire word must appear in the lexicon (in which case it will be found before any stripping is attempted), or [2] the stripping process will fail, and the user-supplied function SPELLING-ERROR will be called. Appendix III contains a suggested collection of attributes.

PUNCTUATION

The program checks for the terminal characters `!', `?', `;', `:', `,' and `.' -- exclamation point, question mark, semicolon, colon, comma and period — immediately after the full form has been checked against the lexicon. (So, for instance, abbreviations like `MR.' may appear in the lexicon.) No check for internal punctuation (like hyphens or dashes) or preceding punctuation (like quotation marks) is made. See Figure 1.

CONTRACTIONS

The contractions ´M, ´RE, ´VE, ´S, N´T, ´LL, and ´D are separated and expanded into their full forms: AM, ARE, HAVE, IS, NOT, WILL, and WOULD -- but only provided that the P.O.S attribute of the expanded form (e.g., HAVE) agrees with one or more senses of the preceding stem (e.g., a PROnoun or a MODAL). The program knows about the exceptional forms CAN´T, WON´T, SHAN´T, and AIN´T (which do not "strip" properly), and attempts to convert these to CAN NOT, WILL NOT, SHALL NOT, and AM NOT, respectively. See Figure 1.

Figure 1
Punctuation, Contractions and -'S

Interpretation: if answer is yes, follow double line (to right), else follow single line down; `end' refers to the last characters in the string; `last' refers to the single last character; `2nd' refers to the next-to-last character; `3rd' refers to the character preceding `2nd'; `word' refers to the current string, as processed — it may not actually be an English word; `cutoff' means to remove the character(s) from the end of the word (but from the front, in prefix-stripping); `add' means to add the character(s) to the end of the current string; `cons' means to add the categories associated with an ending to the ENDings list, for use by TRY; and lastly, `TRY' means to find the current string in the lexicon if possible, to check the senses against the categories in the ENDings list for consistency, and to return the agreeing senses if extant, else fail (answer NIL, meaning `no').

VERBS

The rules for regular verb endings are rather simple: one strips -S, sometimes -ES, -D, sometimes -ED, and -ING; this may involve restoring the original shift from -Y to -I-, adding a deleted -E, un-doubling doubled consonants, or even restoring an -IE to -Y- shift. There are heuristics to minimize the number of false "tries"; for example, knowing that no verbs (or nouns) end in -XE allows one to strip -XES directly to -X, without stripping to -XE, trying that and failing, then stripping to -X. Note that the same cannot be stated of, say, -SES (except in the form -SSES) due to some counter-examples like SURMISE and SURPRISE. Figures 2-3 depict the "regular ending" portion of the program; the reader can extract from them the logic of stripping regular verbs.

Irregular verbs are another matter; there is no "global" set of rules to account for irregularly inflected verbs. While some of the -EN endings are relatively predictable, others are exemplified by very few members of the language. Some -ED and -EN forms involve vowel shifts that, while phonologically predictable, are graphically unpredictable: consider SHOE-SHOD vs. SHOOT-SHOT. Others are not even phonologically predictable: consider SINK-SANK-SUNK vs. SLINK-SLUNK, or STICK-STUCK vs. STRIKE-STRUCK. And even when a "family" of verbs is predictable, it is probably not worth the effort to write the stripping rules if that family numbers, e.g., only two members. In this type of situation, we are faced with an aesthetic decision: whether to code stripping rules, or make "exceptional" lexical entries. We have tended to code stripping rules where the family numbers three or more members, and where there is relatively little chance of confusion with other families. Tables 1-3 exhibit the family types and their members. Figure 3 includes the relatively regular -EN endings. Figure 4 depicts that part of the analyzer devoted to the very irregular verbs.

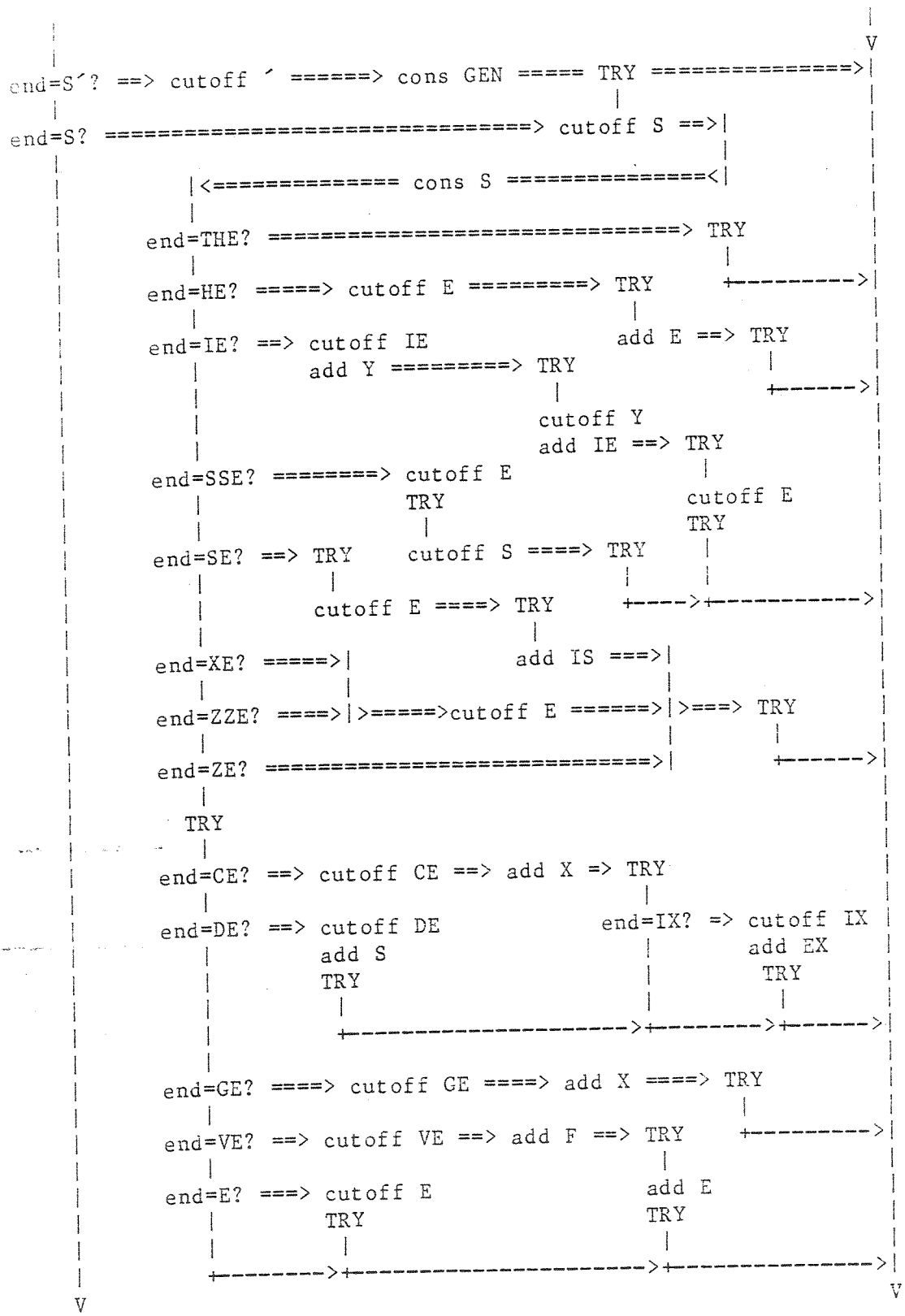

Figure 2
-S´ and -S

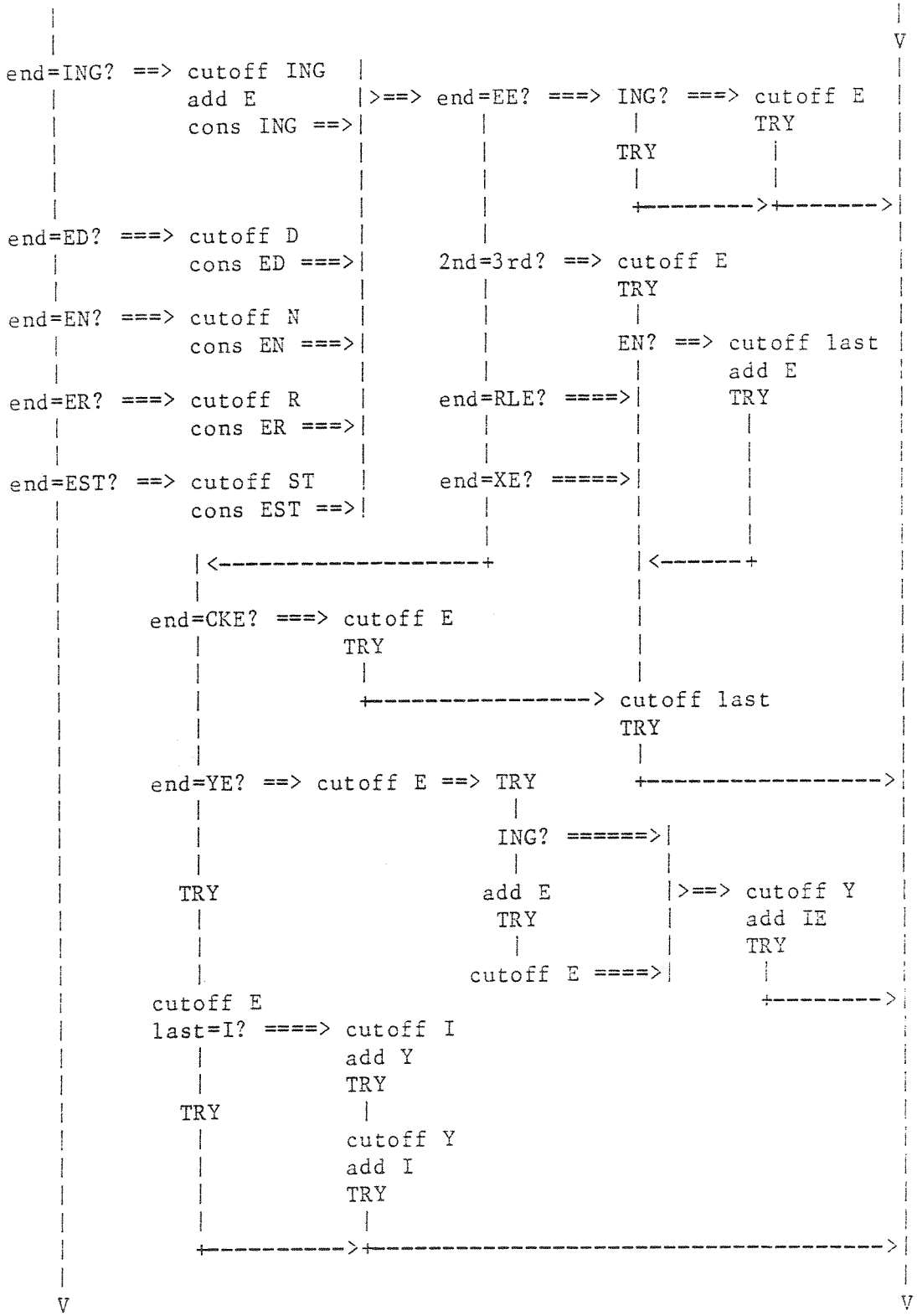

Figure 3
-ING, -ED, -EN, -ER, and -EST

cutoff -T: burnt-burn dealt-deal dreamt-dream leant-lean leapt-leap learnt-learn meant-mean misdealt-misdeal spoilt-spoil sunburnt-sunburn

change -T to -D: bent-bend blent-blend built-build gilt-gild girt-gird lent-lend rent-rend sent-send spent-spend

change -LT to -LL: dwelt-dwell smelt-smell spelt-spell spilt-spill

change -EPT to -EEP: crept-creep kept-keep slept-sleep swept-sweep wept-weep

change -EFT to -EAVE: bereft-bereave cleft-cleave left-leave

Table 1
Rules for -ED inflections ending in -T

change -EN to -E: arisen-arise awaken-awake betaken-betake
driven-drive forgiven-forgive forsaken-forsake given-give
graven-grave laden-lade mistaken-mistake partaken-partake
proven-prove risen-rise riven-rive seen-see shaken-shake
shriven-shrive striven-strive taken-take thriven-thrive
undertaken-undertake waken-wake

change -ccEN to -cE: backbitten-backbite bestridden-bestride
bitten-bite chidden-chide hidden-hide ridden-ride smitten-smite
stridden-stride underwritten-underwrite written-write

change -ccEN to -c: bidden-bid forbidden-forbid

drop -EN: beaten-beat been-be befallen-befall browbeaten-browbeat
eaten-eat fallen-fall

change -WN to -W: blown-blow drawn-draw grown-grow hewn-hew known-know
mown-mow sewn-sew shown-show sown-sow strewn-strew thrown-throw
withdrawn-withdraw

change -UNK to -INK: drunk-drink shrunk-shrink sunk-sink stunk-stink

change -ORN to -EAR: forsworn-forswear shorn-shear sworn-swear
torn-tear worn-wear

Table 2
Rules for unambiguously -EN inflections

change -OKE to -EAK: bespoke-bespeak broke-break spoke-speak
change -OLE to -EAL: stole-steal

change -ORE to -EAR: bore-bear forbore-forbear forswore-forswear
swore-swear tore-tear wore-wear

change -OVE to -EAVE: clove-cleave hove-heave wove-weave

change -OcE to -IcE: abode-abide arose-arise bestrode-bestride
bode-bide dove-dive drove-drive rode-ride rose-rise shone-shine
shrove-shrive smote-smite strode-stride strove-strive
throve-thrive underwrote-underwrite wrote-write

change -UNG to -ING: clung-cling flung-fling hamstrung-hamstring
slung-sling stung-sting strung-string swung-swing wrung-wring

change -ANG to -ING: rang-ring sang-sing sprang-spring

change -ANK to -INK: drank-drink shrank-shrink sank-sink stank-stink

change -EW to -OW: blew-blow crew-crow grew-grow knew-know threw-throw

change -AID to -AY: gainsaid-gainsay inlaid-inlay laid-lay
mislaid-mislay paid-pay said-say waylaid-waylay

change -OUND to -IND: bound-bind found-find ground-grind wound-wind

change -OOK to -AKE: betook-betake forsook-forsake mistook-mistake
shook-shake took-take undertook-undertake

Table 3
Rules for -ED inflections with internal vowel shifts

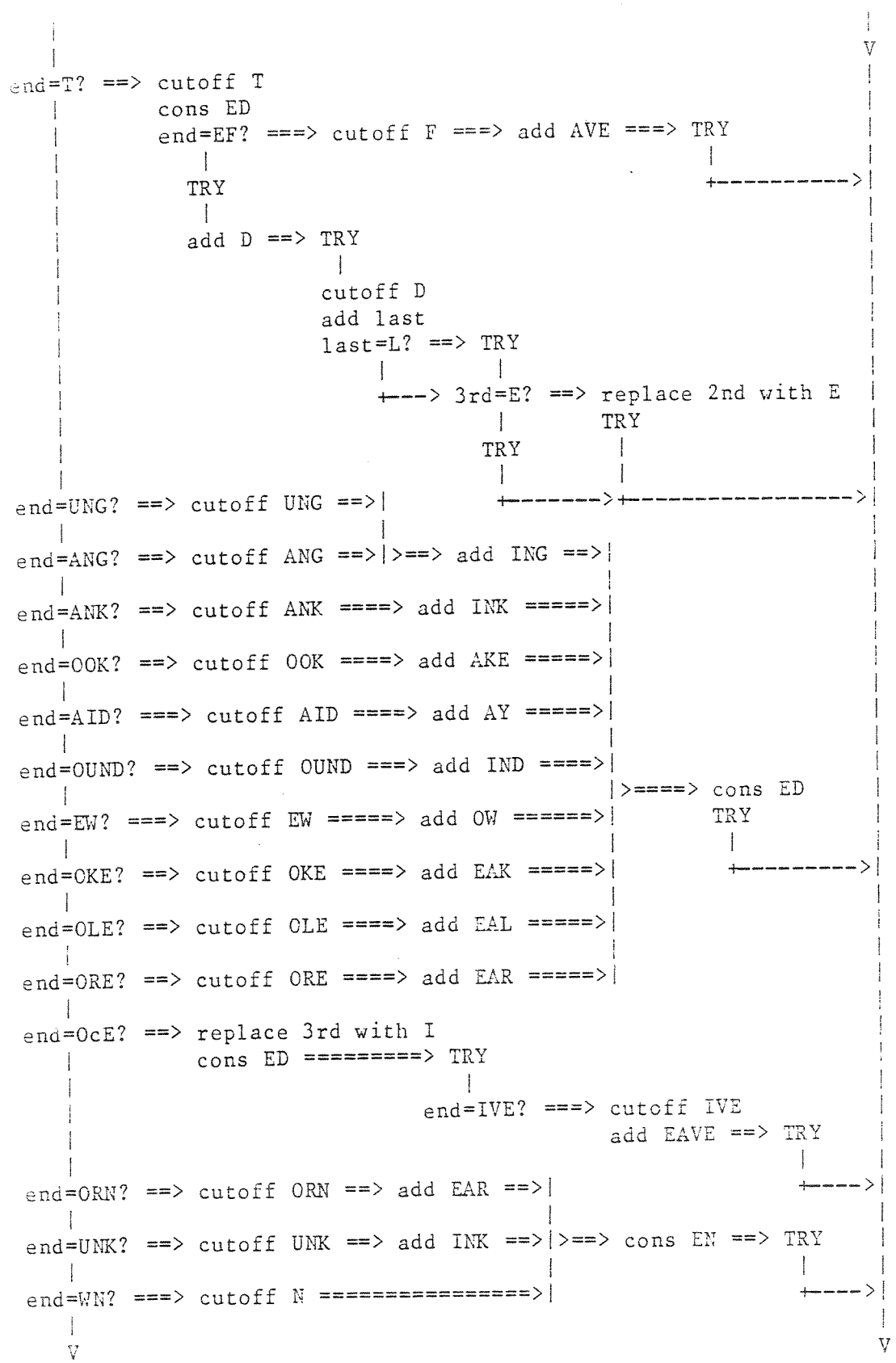

Figure 4
Verbs with internal vowel shifts

NOUNS

The rules for regular noun endings are quite simple: one strips -S, sometimes -ES, -'S, -S', and sometimes -ES'. Actually, the regular plural inflection of a noun involves stripping two suffixes -- the genitive "'", and (separately) the plural `(E)S'. (Thus FATHERS' -> FATHER S GEN.) See Figures 1-2.

Irregular plural inflections are more complicated; some forms borrowed from Latin or Greek retain their native inflections. Thus we have cases like ALUMNUS-ALUMNI, FORMULA-FORMULAE, QUANTUM-QUANTA, GENUS-GENERA from Latin, and INDEX-INDICES, APPENDIX-APPENDICES, SCHEMA-SCHEMATA, PHENOMENON-PHENOMENA, NOMEN-NOMINA, APHIS-APHIDES, PHALANX-PHALANGES from Greek. Where the inflected forms end with -A, the root-determination is non-deterministic: the analyzer must try one and, failing, try another, until it hits the right root. Figures 2 and 5 include the portions of the flowchart devoted to these cases.

ADJECTIVES AND ADVERBS

The rules for adjectives and adverbs are quite simple: one strips -R, sometimes -ER, -ST, sometimes -EST, and the adverbial suffix -LY. Sometimes this may require reversing the -Y to -I- shift, etc. Of the very few irregular adjective formations (e.g., GOOD, BETTER, BEST), none are amenable to stripping rules; hence all must be entered as lexical exceptions. See Figures 3 and 5.

ORDINAL NUMBERS

The rules for ordinal suffix removal are straightforward; the only special cases are in spelled-out numbers: reversing the -Y to -IE- conversion (e.g., TWENTIETH -> TWENTY TH), adding a deleted -E (e.g., NINTH -> NINE TH), reversing a voiced-voiceless shift (e.g., TWELFTH -> TWELVE TH), or adding a deleted -T (only instance: EIGHTH -> EIGHT TH). In the case of ordinal numbers (e.g., 1st, 2nd, 3rd, 4th), -ST, -ND, and -RD are all treated like -TH, and the user function LEXENTRIES is called to construct (or find) a lexical entry for the numeral. See Figure 6.

PREFIXES

The only prefixes removed are those which indicate a simple negative inversion: NON-, UN-, IN-, IM-, and Icc- (I followed by a doubled consonant, e.g., IRREGULAR). All are converted to the canonical NEG. Prefix stripping is only tried after all suffix rules have failed to produce a root; if applicable, the prefix rules recursively invoke the analyzer with the remainder of the word, in an attempt to find the root. Only if this succeeds will the prefix be removed; otherwise, the entire stripping process fails, and the user function SPELLING-ERROR is called. See Figure 6.

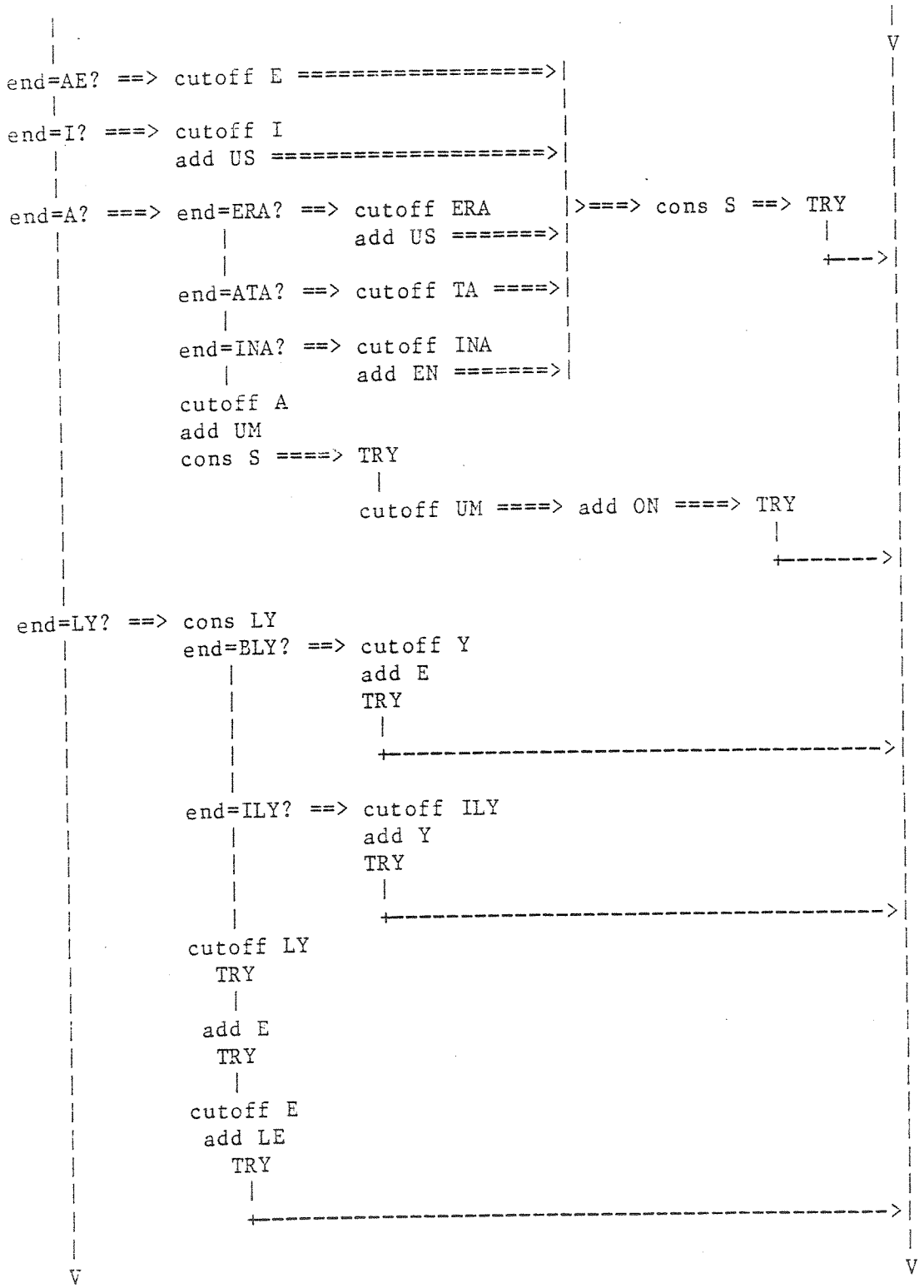

Figure 5
Foreign plural nouns and -LY

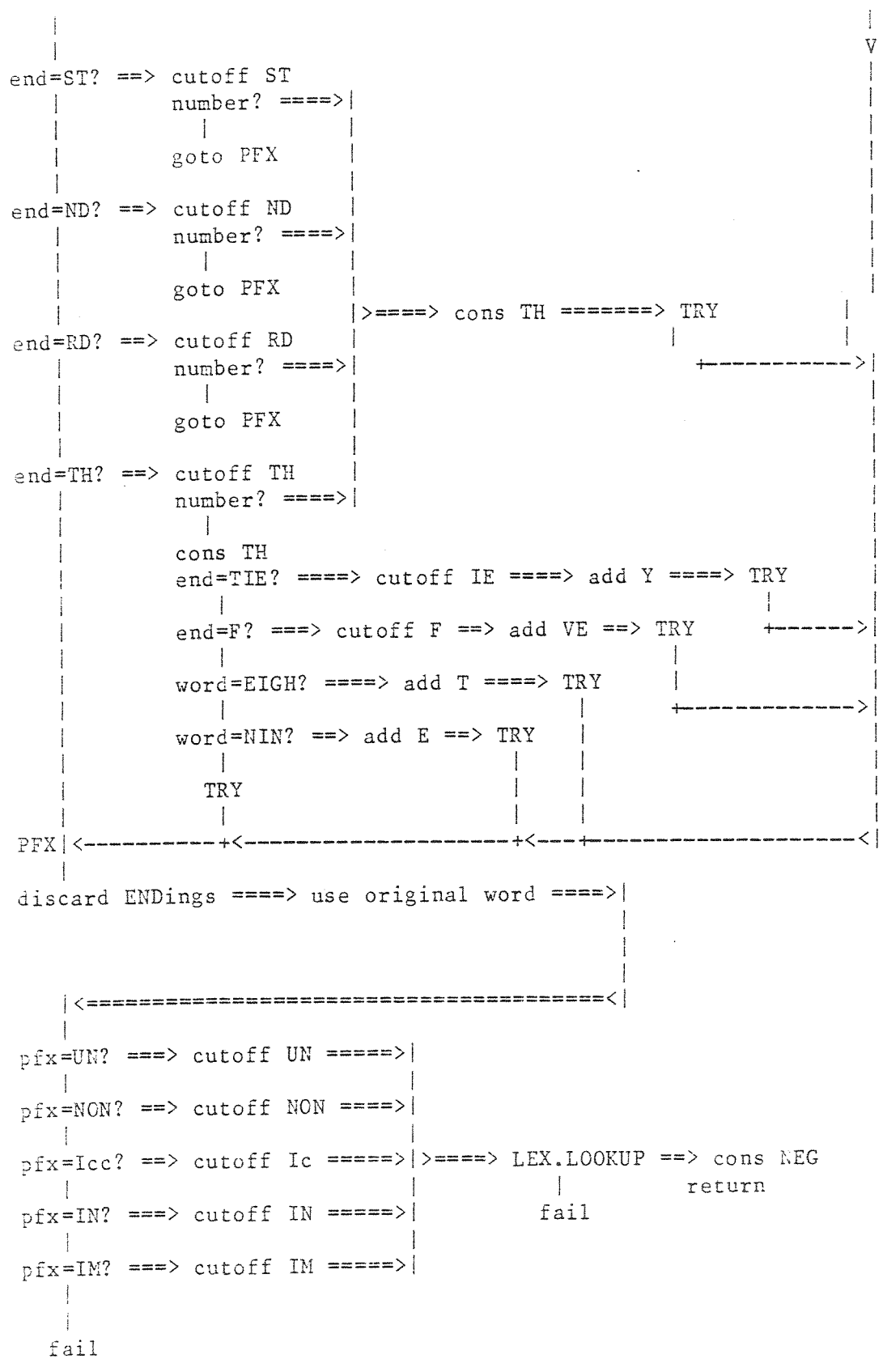

Figure 6
Ordinal numbers and prefixes

Appendix I

The irregular English verbs

(Note: some of the infinitives may also have regular inflections.)

(abide abode)
(arise arose arisen)
(awake awoke awaken)
(backbite backbit backbitten)
(backslide backslid)
(be been being am are is was were)
(bear bore borne)
(beat beaten)
(become became)
(befall befell befallen)
(beget begot begotten)
(begin began begun)
(behold beheld)
(bend bent)
(bereave bereft)
(beseech besought)
beset
(bespeak bespoke bespoken)
(bestride bestrode bestridden)
bet
(betake betook betaken)
(bethink bethought)
(bid bade bidden)
(bide bode)
(bind bound)
(bite bit bitten)
(bleed bled)
(blend blent)
(bless blest)
(blow blew blown)
(break broke broken)
(breed bred)
(bring brought)
broadcast
(browbeat browbeaten)
(build built)
(burn burnt)
burst
(buy bought)
cast
(catch caught)
(chide chid chidden)
(choose chose chosen)
(cleave cleft clove cloven)
(cling clung)
(clothe clad)
(come came)
cost
(creep crept)
(crow crew)
cut
(deal dealt)
(dig dug)
(dive dove)
(do did done)
(draw drew drawn)
(dream dreamt)
(drink drank drunk)
(drive drove driven)
(dwell dwelt)
(eat ate eaten)
(fall fell fallen)
(feed fed)
(feel felt)
(fight fought)
(find found)
fit
(flee fled)
(fling flung)
(fly flew flown)
(forbear forbore forborne)
(forbid forbade forbidden)
forecast
(forget forgot forgotten)
(forsake forsook forsaken)
(forswear forswore forsworne)
(freeze froze frozen)
(gainsay gainsaid)
(get got gotten)
(gild gilt)
(gird girt)
(give gave given)
(go went gone)
(grave graven)
(grind ground)
(grow grew grown)
(hamstring hamstrung)
(hang hung)
(have had has)
(hear heard)
(heave hove)
(hew hewn)
(hide hid hidden)
hit
(hold held)
hurt
(inlay inlaid)
(keep kept)
(kneel knelt)
knit
(know knew known)
(lade laden)
(lay laid)
(lead led)
(lean leant)
(leap leapt)

(learn learnt)
(leave left)
(lend lent)
let
(lie lay lain)
(light lit)
(lose lost)
(make made)
(mean meant)
(meet met)
(misdeal misdealt)
(mislay mislaid)
(mislead misled)
(mistake mistook mistaken)
(misunderstand misunderstood)
(mow mown)
(partake partook partaken)
(pay paid)
(plead pled)
(prove proven)
put
quit
read
(rend rent)
rid
(ride rode ridden)
(ring rang rung)
(rise rose risen)
(run ran)
(say said)
(see saw seen)
(seek sought)
(sell sold)
(send sent)
set
(sew sewn)
(shake shook shaken)
(shear shorn)
shed
(shine shone)
(shoe shod)
(shoot shot)
(show shown)
(shrink shrank shrunk)
(shrive shrove shriven)
shut
(sing sang sung)
(sink sank sunk)
(sit sat)
(slay slew slain)
(sleep slept)
(slide slid)
(sling slung)
(slink slunk)
slit
(smell smelt)
(smite smote smitten)
(sow sown)
(speak spoke spoken)
(speed sped)
(spell spelt)
(spend spent)
(spill spilt)
(spin spun)
(spit spat)
split
(spoil spoilt)
spread
(spring sprang sprung)
(stand stood)
(stave stove)
(steal stole stolen)
(stick stuck)
(sting stung)
(stink stank stunk)
(strew strewn)
(stride strode stridden)
(strike struck stricken)
(string strung)
(strive strove striven)
(sunburn sunburnt)
(swear swore sworn)
sweat
(sweep swept)
(swell swollen)
(swim swam swum)
(swing swung)
(take took taken)
(teach taught)
(tear tore torn)
(tell told)
(think thought)
(thrive throve thriven)
(throw threw thrown)
thrust
(tread trod trodden)
(undergo underwent undergone)
(understand understood)
(undertake undertook undertaken)
(underwrite underwrote underwritten)
upset
(wake woke waken)
(waylay waylaid)
(wear wore worn)
(weave wove woven)
wed
(weep wept)
wet
(win won)
(wind wound)
(withdraw withdrew withdrawn)
(withhold withheld)
(withstand withstood)
(work wrought)
(wring wrung)
(write wrote written)

Appendix II

Irregular verb forms not handled by the Analyzer

am
are
ate
awoke
backbit
backslid
bade
became
been
befell
began
begot
begotten
begun
beheld
besought
bespoken
bethought
bit
bled
blest
borne
bought
bred
broken
brought
came
caught
chid
chose
chosen
clad
cloven
did
done
drew
dug
fed
fell
fled
flew
flown
forbade
forborne
forgot
forgotten
forsworne
fought
froze
frozen
gave
gone
got
gotten
had
has
heard
held
hid
hung
is
lain
lay
led
lit
lost
made
met
misled
misunderstood
pled
ran
sat
saw
shod
shot
slain
slew
slid
sold
sought
spat
sped
spoken
spun
stolen
stood
stove
stricken
struck
stuck
swam
swollen
swum
taught
thought
told
trod
trodden
undergone
understood
underwent
was
went
were
withdrew
withheld
withstood
woke
won
woven
wrought

Appendix III
Suggested AFFIXLIST associations

ED	(V)
EN	(V)
ER	(ADJ)
EST	(ADJ)
GEN	(N PRO)
ING	(V BE DO HAVE)
LY	(ADJ)
NEG	(N V ADJ ADV)
-N	(N)
-S	(N V DO NUM)
TH	(NUM)
AM	(PRO)
ARE	(PRO)
HAVE	(PRO MODAL)
IS	(N PRO)
NOT	(BE DO HAVE MODAL)
WILL	(PRO N)
WOULD	(PRO N)

Appendix IV
The INTERLISP program

```
(LEX.LOOKUP                                                    (*
  [LAMBDA (W POS)

"     LEX.LOOKUP is the top-level function.  W (the word to be
analyzed) must be in upper case; POS must be a list of part-of-speech
(grammatical category) restrictions on the root of W, or a single
part-of-speech, else NIL if there are no restrictions.  LEX.LOOKUP
returns a list of all possible analyses of W (within the category
restrictions, if any), else NIL if there are none; each analysis is
(in this implementation) a list of `morphemes´ comprising the word W.
LEX.LOOKUP does NOT correct spelling errors; the author has another
lexical analysis program that does, operating on quite different
principles, and which is language-independent; the dictionary
required, however, is more extensive.")

    (OR (NULL POS)
      (LISTP POS)
      (SETQ POS (LIST POS)))
    (PROG (END X (STRIPS (LEXENTRIES W POS)))   (* (TRY))
          [COND
          ((NOT (LITATOM W)))
          [[SETQ END (CAR (FMEMB (NTHCHAR W -1)
                                 (QUOTE (? ! ; : , %.]
                                                (* punctuation?)

            (COND
              ((SETQ X (LEX.LOOKUP (MKATOM (SUBSTRING W 1 -2))
                                   POS))

                (LEXEND X)
                (SETQ STRIPS (NCONC STRIPS X]
           ([SETQ X (WDSTRIP (DREVERSE (UNPACK W]
             (SETQ STRIPS (NCONC STRIPS X]
           (RETURN (INTERSECTION STRIPS STRIPS])

(WDSTRIP
  [LAMBDA (L)
    (PROG (CATS END X)
          (AND
          (SETQ X (FMEMB (QUOTE ´)
                         L))                    (* -´-)
          (SETQ L (PROG1 (CDR X)
                         (FRPLACD X)
                         (SETQ X L)))

          (OR
            (SETLEXCATS
              (SELECTQ
                (PACK (DREVERSE X))
                [´T
                  (COND
                    ([SETQ X
                       (REVERSE
```

```
                (CDR (FASSOC W (QUOTE ((CAN'T  C A N)
                                       (WON'T  W I L L)
                                       (SHAN'T S H A L L)
                                       (AIN'T  A M]
            (SETQ L X)
            (QUOTE NOT))
          ((EQ (CAR L)
               (QUOTE N))
            (AND
              (SETLEXCATSQ NOT)
              (COND
                ((SETQ X
                     (LEX.LOOKUP (PACK (DREVERSE (CDR L)))
                                 CATS))
                  (LEXEND X)
                  (RETURN X]
      ('S (AND (SETLEXCATSQ -S)
               (WDTRY L))
          (AND (SETLEXCATSQ IS)
               (WDTRY L))
          (QUOTE GEN))
      ('RE (QUOTE ARE))
      ('VE (QUOTE HAVE))
      ('LL (QUOTE WILL))
      ('M (QUOTE AM))
      ('D (QUOTE WOULD))
      [' (AND (EQ (CAR L)
                  (QUOTE S))
              (SETLEXCATSQ GEN)
              (COND
                ((WDTRY L)
                  (RETURN))
                ((SETQ X (WDSTRIPS L CATS))
                  (LEXEND X)
                  (RETURN X]
      NIL))
  (GO ERR))
(GO TRY))
(OR (SETQ X (CDDR L))
  (GO ERR))
(SELECTQ
(CAR L)
(S (AND (SETQ X (WDSTRIPS L POS))
        (RETURN X))                    (* -S)
   (GO PFX))
[(D G N R T H)
  (COND
    [(SETLEXCATS (SELECTQ
                   (CAR L)
                   (G                  (* -ING)
                      (AND (EQ (CADR L)
                               (QUOTE N))
                           (EQ (CAR X)
                               (QUOTE I))
```

```
                    (CDDR X)
                    (SETQ L X)
                    (FRPLACA L (QUOTE E))
                    (QUOTE ING)))
              [(D N R)          (* -ED -EN -ER)
                (AND (EQ (CADR L)
                         (QUOTE E))
                     (CDR X)
                     (PROG1 (PACK* (QUOTE E)
                                   (CAR L))
                            (SETQ L (CDR L]
              (T                 (* -EST)
                 (AND (EQ (CADR L)
                          (QUOTE S))
                      (EQ (CAR X)
                          (QUOTE E))
                      (CDDR X)
                      (SETQ L X)
                      (QUOTE EST)))
              NIL))
(COND
  [(EQ (CAR L)
       (CADR L))                  (* -EE-)
    (AND (EQ END (QUOTE ING))
         (SETQ L (CDR L]
  [(EQ (CADR L)
       (CADDR L))                 (* -11-)
    (SETQ L (CDR L))
    (COND
      ((WDTRY L)
        (GO PFX))
      ([AND (EQ END (QUOTE EN))
            (WDTRY (FRPLACA L (QUOTE E]
                               (* -ccEN)
        (GO PFX))
      ((SETQ L (CDR L]
   ((AND (EQ (CADR L)
             (QUOTE L))
         (EQ (CADDR L)
             (QUOTE R)))          (* -RL-)
     (SETQ L (CDR L)))
   ((EQ (CADR L)
        (QUOTE Y))                (* -Y-)
     (AND (COND
            ((WDTRY (CDR L)))
            ((NEQ END (QUOTE ING))
              (WDTRY L)))
          (GO PFX))
     (FRPLACA (CDR L)
              (QUOTE I)))
   ((EQ (CADR L)
        (QUOTE X))                (* -X-)
     (SETQ L (CDR L)))
```

```
    ((AND (EQ (CADR L)
              (QUOTE K))
          (EQ (CADDR L)
              (QUOTE C))
          (SETQ L (CDR L))        (* -CK-)
          NIL))
     ((WDTRY L)
       (GO PFX))
     ((EQ (CAR (SETQ L (CDR L)))
          (QUOTE I))              (* -I-)
       (COND
         ((WDTRY (FRPLACA L (QUOTE Y)))
           (GO PFX))
         ((FRPLACA L (QUOTE I]
((EVERY X (FUNCTION SMALLP))      (* ordinal?)
  (AND (SELECTQ (CAR L)
                (H                (* nTH)
                   (EQ T (CADR L)))
                [D                (* 3RD 2ND)
                   (FMEMB (CADR L)
                          (QUOTE (N R]
                (T                (* 1ST)
                   (EQ (CADR L)
                       (QUOTE S)))
                NIL)
       (SETLEXCATSQ TH)
       (WDTRY L)
       (RETURN))
  (GO ERR))
 ((SELECTQ
     (CAR L)
     [N (OR [AND (SETLEXCATSQ EN)
                 (COND
                   ((EQ (CADR L)
                        (QUOTE W))
                                  (* -WN)
                     (SETQ L (CDR L)))
                   ((AND (EQ (CADR L)
                             (QUOTE R))
                         (EQ (CAR X)
                             (QUOTE O)))
                                  (* -ORN)
                     (FRPLACA X (QUOTE E))
                     (FRPLACA L (QUOTE R))
                     (FRPLACA (CDR L)
                              (QUOTE A]
            (AND (EQ (CADR L)
                     (QUOTE A))
                 (SETLEXCATSQ -N)
                 (COND
                   ((WDTRY (CDR L))
                                  (* -N)
                     (GO PFX))
```

```
                ((WDTRY X)     (* -AN)
                  (GO PFX))
                ((WDTRY (FRPLACA (CDR L)
                                 (QUOTE O)))
                  (GO PFX))
                ((NEQ (CAR X)
                      (QUOTE I))
                  NIL)
                ((WDTRY (CDR X))
                                (* -IAN)
                  (GO PFX))
                ((SETQ L (FRPLACA X (QUOTE Y]
[T
  (AND
    (SETLEXCATSQ ED)
    (COND
      ((FMEMB (CADR L)
              (QUOTE (L N P M R)))
                              (* -cT)
        (AND (COND
               ((WDTRY (CDR L)))
               [(WDTRY (FRPLACA L (QUOTE D]
               ((AND (EQ (CAR (FRPLACA L
                                       (CADR L)))
                         (QUOTE L))
                              (* -LT)
                    (WDTRY L)))
               ((EQ (CADDR L)
                    (QUOTE E))
                              (* -EcT)
                 (FRPLACA (CDR L)
                          (QUOTE E))
                 (WDTRY L)))
             (GO PFX)))
      ((AND (EQ (CADR L)
                (QUOTE F))
            (EQ (CAR X)
                (QUOTE E))) (* -EFT)
        (FRPLACA (CDR L)
                 (QUOTE A))
        (ATTACH (QUOTE E)
                (FRPLACA L (QUOTE V]
[G                              (* -ANG -UNG)
   (AND (EQ (CADR L)
            (QUOTE N))
        (FMEMB (CAR X)
               (QUOTE (U A)))
        (SETLEXCATSQ ED)
        (FRPLACA X (QUOTE I]
[D (COND
     [[AND (EQ (CADR L)
               (QUOTE N))
           (EQ (CAR X)
               (QUOTE U))
```

```
                    (EQ (CADR X)
                        (QUOTE O))
                    (SETLEXCATSQ ED] (* -OUND)
                  (FRPLACD X (CDDR X))
                  (FRPLACA X (QUOTE I]
               ([AND (EQ (CADR L)
                         (QUOTE I))
                     (EQ (CAR X)
                         (QUOTE A))
                     (SETLEXCATSQ ED] (* -AID)
                  (SETQ L (CDR L))
                  (FRPLACA L (QUOTE Y]
          [H                                (* -TH)
             (AND (EQ T (CADR L))
                  (SETLEXCATSQ TH)
                  (COND
                    ((EQ W (QUOTE EIGHTH))
                      (SETQ L (CDR L)))
                    [(EQ W (QUOTE NINTH))
                      (SETQ L (FRPLACA (CDR L)
                                       (QUOTE E]
                    [(FMEMB W (QUOTE (FIFTH TWELFTH)))
                      (FRPLACA X (QUOTE V))
                      (SETQ L (FRPLACA (CDR L)
                                       (QUOTE E]
                    [(AND (CDDR (CDDDR X))
                          (EQ (CAR X)
                              (QUOTE E))
                          (EQ (CADR X)
                              (QUOTE I))
                          (EQ T (CADDR X)))
                                       (* -TIETH)
                      (SETQ L (FRPLACA (CDR X)
                                       (QUOTE Y]
                    ((SETQ L X]
             NIL))
       ((GO PFX]
  [E (COND
       [(AND (EQ (CAR X)
                 (QUOTE O))
             (SETLEXCATSQ ED)
             (SELECTQ (CADR L)
                      [V                (* -OVE)
                         (FRPLACA X (QUOTE I))
                         (AND (WDTRY L)
                              (GO PFX))
                         (ATTACH (QUOTE A)
                                 (FRPLACA (CDDR L)
                                          (QUOTE E]
                      ((D T S N)        (* -OcE)
                        (FRPLACA X (QUOTE I)))
                      (R                (* -ORE)
                         (FRPLACA X (QUOTE E))
```

```
                        (FRPLACA L (QUOTE R))
                        (FRPLACA (CDR L)
                                 (QUOTE A)))
                    ((K L)          (* -OKE -OLE)
                      (FRPLACA X (QUOTE E))
                      (FRPLACA L (CADR L))
                      (FRPLACA (CDR L)
                               (QUOTE A)))
                    (SETQ END]
     ((AND (EQ (CADR L)
               (QUOTE A))
           (SETLEXCATSQ -S))        (* -AE)
       (SETQ L (CDR L)))
     [(AND (EQ (CADR L)
               (QUOTE S))
           (EQ (CAR X)
               (QUOTE E))
           (SETLEXCATSQ -N))        (* -ESE)
       (COND
         ((WDTRY (CDR X))
           (GO PFX))
         ((WDTRY (FRPLACA X (QUOTE A)))
           (GO PFX))
         ((SETQ L (CDDR X]
     ((GO PFX]
[Y (COND
     [(AND (EQ (CADR L)
               (QUOTE L))
           (SETLEXCATSQ LY))        (* -LY)
       (COND
         [(EQ (CAR X)
              (QUOTE I))            (* -ILY)
           (SETQ L (FRPLACA X (QUOTE Y]
         ((EQ (CAR X)
              (QUOTE B))            (* -BLY)
           (FRPLACA L (QUOTE E)))
         ((WDTRY X)
           (GO PFX))
         ((WDTRY (CONS (QUOTE E)
                       (CDDR L)))
           (GO PFX))
         ((FRPLACA L (QUOTE E]
     ((GO PFX]
[K (COND
     ([AND (EQ (CADR L)
               (QUOTE N))
           (COND
             ((EQ (CAR X)
                  (QUOTE A))        (* -ANK)
               (SETLEXCATSQ ED))
             ((EQ (CAR X)
                  (QUOTE U))        (* -UNK)
               (SETLEXCATSQ EN]
```

```
            (FRPLACA X (QUOTE I)))
          ((AND (EQ (CADR L)
                    (QUOTE O))
                (EQ (CAR X)
                    (QUOTE O))
                (SETLEXCATSQ ED))          (* -OOK)
            (FRPLACA X (QUOTE A))
            (FRPLACA L (QUOTE E))
            (FRPLACA (CDR L)
                     (QUOTE K)))
          ((GO PFX]
      [W (COND
          ((AND (EQ (CADR L)
                    (QUOTE E))
                (SETLEXCATSQ ED))          (* -EW)
            (FRPLACA (CDR L)
                     (QUOTE O)))
          ((GO PFX]
      [A (OR (SETLEXCATSQ -S)
             (GO PFX))                     (* -A)
         (COND
           ((AND (EQ (CADR L)
                     (QUOTE R))
                 (EQ (CAR X)
                     (QUOTE E)))           (* -ERA)
            (SETQ L (CDR L))
            (FRPLACA X (QUOTE U))
            (FRPLACA L (QUOTE S)))
           ((AND (EQ T (CADR L))
                 (EQ (CAR X)
                     (QUOTE A)))           (* -ATA)
            (SETQ L X))
           ((AND (EQ (CADR L)
                     (QUOTE N))
                 (EQ (CAR X)
                     (QUOTE I)))           (* -INA)
            (SETQ L (CDR L))
            (FRPLACA X (QUOTE E)))
           ([WDTRY (ATTACH (QUOTE M)
                           (FRPLACA L (QUOTE U]
            (GO PFX))
           ((FRPLACA L (QUOTE N))
            (FRPLACA (CDR L)
                     (QUOTE O]
      [I (AND (SETLEXCATSQ -N)
              (WDTRY (CDR L)))             (* -I)
         (OR (SETLEXCATSQ -S)
             (GO PFX))
         (ATTACH (QUOTE S)
                 (FRPLACA L (QUOTE U]
      (GO PFX))
TRY (WDTRY L)                               (* (TRY))
PFX (SETQQ X 3)                             (* try PREFIXES)
```

```
          (AND (SETLEXCATSQ NEG)
               (ILESSP 4 (NCHARS W))
               (SELECTQ (NTHCHAR W 1)
                        (U                          (* UN-)
                           (EQ (NTHCHAR W 2)
                               (QUOTE N)))
                        [N                          (* NON-)
                           (AND (EQ (NTHCHAR W 2)
                                    (QUOTE O))
                                (EQ (NTHCHAR W 3)
                                    (QUOTE N))
                                (COND
                                  ((EQ (NTHCHAR W 4)
                                       (QUOTE -))
                                    (SETQQ X 5))
                                  ((SETQQ X 4]
                        [I                          (* Icc- IM- IN-)
                           (OR (EQ (NTHCHAR W 2)
                                   (NTHCHAR W 3))
                               (FMEMB (NTHCHAR W 2)
                                      (QUOTE (M N]
                        NIL)
               (SETQ X (LEX.LOOKUP (MKATOM (SUBSTRING W X -1))
                                   CATS))
               (LEXBEG X (QUOTE NEG))
               (RETURN X))
         ERR (OR STRIPS (SPELLING-ERROR W END))
             (RETURN])

(WDSTRIPS
  [LAMBDA (L POS STRIPS)
    (PROG (X CATS END)                              (* -S)
          (OR (SETLEXCATSQ -S)
            (RETURN))
          (SETQ L (CDR L))
          [AND
          (EQ (CAR L)
              (QUOTE E))                            (* -ES)
          (SELECTQ
            (CADR L)
            [H                                      (* -HES)
               (COND
                 ((EQ T (CADDR L)))
                 ((WDTRY (CDR L))
                   (RETURN]
            [I                                      (* -IES)
               (COND
                 ((WDTRY (CONS (QUOTE Y)
                               (CDDR L)))
                   (RETURN))
                 ((WDTRY L)
                   (RETURN))
                 ((SETQ L (CDR L]
```

```
            [S                                        (* -SES)
               (COND
                 ((AND (EQ (CADDR L)
                           (QUOTE S))
                       (SETQ L (CDR L))              (* -SSES)
                       NIL))
                 ((WDTRY L)
                   (RETURN))
                 ((WDTRY (CDR L))
                   (RETURN))
                 ((FRPLACA L (QUOTE I))
                   (ATTACH (QUOTE S)
                           L]
            (X                                        (* -XES)
               (SETQ L (CDR L)))
            [Z                                        (* -ZES)
               (AND (EQ (CADDR L)
                        (QUOTE Z))                    (* -ZZES)
                    (SETQ L (CDR L]
            (COND
              ((WDTRY L)
                (SETQ L (CDR L)))
              ((SELECTQ
                  (CADR L)
                  [C                                  (* -CES)
                     (COND
                       ((WDTRY (FRPLACA (FRPLACD L (CDDR L))
                                        (QUOTE X)))
                         (RETURN))
                       ((EQ (CADR L)
                            (QUOTE I))
                         (FRPLACA (CDR L)
                                  (QUOTE E]
                  (D                                  (* -DES)
                     (SETQ L (CDR L))
                     (FRPLACA L (QUOTE S)))
                  (G                                  (* -GES)
                     (SETQ L (CDR L))
                     (FRPLACA L (QUOTE X)))
                  (V                                  (* -VES)
                     (AND (WDTRY (FRPLACA (CDR L)
                                          (QUOTE F)))
                          (RETURN)))
                  (SETQ L (CDR L]
          (WDTRY L))
    STRIPS])

(WDTRY
  [LAMBDA (L)
    (PROG (W X)
          (SETQ L (DREVERSE L))
          (SETQ W (PACK L))
          (DREVERSE L)
```

```
        [COND
        ((SETQ X (LEXENTRIES W CATS))
          (LEXEND X)
          (SETQ STRIPS (NCONC STRIPS X]
        (RETURN X])


(SETLEXCATS
  [LAMBDA (E)
    (COND
      ((NULL (SETQ END E))
      NIL)
      ((NULL (SETQ CATS (FASSOC E AFFIXLIST)))
      NIL)
      ((NULL (SETQ CATS (CDR CATS)))
      (SETQ CATS POS)
      T)
      ((NULL POS))
      ((SETQ CATS (INTERSECTION CATS POS])

(SETLEXCATSQ
  [NLAMBDA (E)
    (SETLEXCATS E])

(SPELLING-ERROR
  [LAMBDA (W AFX)                                    (* user-definable)
    (*  This function is called  when the analyzer cannot discover any
      analysis for  a word;  the user  may define  this function  as
      desired, but LEX.LOOKUP will return NIL in any case.  W is the
      word, and AFX is  the analyzer's most  recent guess about  the
      possible affix.)
    NIL])

(LEXBEG
  [LAMBDA (X PFX)                                    (* user-definable)
    (*  The default  definition of  LEXBEG sticks the  PreFiX onto the
      front of each of the analyses in the list X.)
    (MAPC X (FUNCTION (LAMBDA (A) (ATTACH PFX A])

(LEXEND
  [LAMBDA (X)                                        (* user-definable)
    (*  The default definition of LEXEND sticks the suffix END onto the
      end of each of the analyses in the list X.  Exceptions:  if END
      is in the list DROPSUFFIXES, it  is ignored;  if END is in  the
      list INVERTSUFFIXES, it is treated as a prefix. If this doesn't
      make sense to the reader, don't worry about it.)
    (COND
      ((FMEMB END DROPSUFFIXES))
      ((FMEMB END INVERTSUFFIXES)
      (LEXBEG X END))
      ((MAPC X (FUNCTION (LAMBDA (A) (NCONC1 A END])
```

```
(LEXENTRIES
  [LAMBDA (W C)                                    (* user-definable)

(* " LEXENTRIES is the function that actually looks up words in the
lexicon.  W is  the word or  number, and  C is a  list of  grammatical
category restrictions on the definitions of W.  The value returned  by
LEXENTRIES must be a  list of analyses; each  analysis must be a  list
which  LEXBEG  and   LEXEND  may  alter   destructively  --  hence   a
newly-created list.  (The contents  are arbitrary, but are  presumably
related to W in some way.)  The value may be as simple as: ((Word)).
      Since no definition of LEXENTRIES  would likely suffice for  a
new implementation, no realistic default  is provided.  That which  is
provided is  for illustrative  purposes only,  indicating the  minimal
structure of a result; specifically, it accesses no dictionary, though
such access would naturally be expected in the implementation.")

    (LIST (LIST W])
```

Appendix V
Default Declarations

```
(SETQQ AFFIXLIST ((-S N V DO NUM)
                  (ING V DO HAVE BE)
                  (ED V)
                  (EN V)
                  (ER ADJ)
                  (EST ADJ)
                  (LY ADJ)
                  (GEN N PRO)
                  (NOT DO HAVE BE MODAL)
                  (NEG N V ADJ ADV)
                  (AM PRO)
                  (ARE PRO)
                  (IS N PRO)
                  (HAVE PRO MODAL)
                  (WILL N PRO)
                  (WOULD N PRO)
                  (TH NUM)
                  (-N N)))

(SETQQ CONTRACTIONS ('M 'RE 'VE 'S N'T 'LL 'D))

(SETQQ DROPSUFFIXES NIL)

(SETQQ INVERTSUFFIXES NIL)

(SETQQ PREFIXES (Icc IM IN NON UN))

(SETQQ SUFFIXES (ED EN ER EST GEN ING LY -N NOT -S TH))
```