A TWO-LEVEL MORPHOLOGICAL ANALYSIS OF ENGLISH

Lauri Karttunen and Kent Wittenburg

Appendices ENGLISH DICTIONARY and ENGLISH AUTOMATA contain a small dictionary and a grammar for English. They are designed to serve as input to Kimmo Koskenniemi's Two-level morphological analyzer/generator. The program generates and recognizes all (and only) well-formed inflected forms of the words in the dictionary.

1. Dictionary

ENGLISH DICTIONARY begins with a list of "alternations." These correspondences indicate which lexicons may be accessed in order to continue with a given root form. This aspect of the program ensures that a noun and a verb reading for spies will be analyzed, but only the verb interpretation of spied.

The major "categories" are indicated by a slash before the alphabetic designation, e.g., /V stands for regular verbs. The present system for morphosyntax requires separate categories for irregular forms; otherwise, nothing would prevent an irregular root such as have from being followed with any and all regular suffixes. Forms such as *haves or *haved would then be permitted. All categories which begin with /I are irregular categories which have a more restricted set of lexicon continuations than the regular forms. The irregular forms which in a sense fill in the gaps left by the lack of a full morphological paradigm are at present simply listed in the Root lexicon. To give a complete example, the verb have is assigned to /IV2, which indicates that all regular verb forms except third-person singular (P3), past (PS), and past participle (PPS) may attach to the root. These three irregular forms are then listed in the Root lexicon with the symbol # as continuation class, indicating that no suffixes may follow.

The rest of the dictionary file consists of a list of lexicons

with each entry in each lexicon consisting of three elements.[1] The analysis of English nouns in ENGLISH DICTIONARY required four suffix lexicons. The result is that count nouns and mass nouns, as well as irregular singular and plural forms, all combine in the expected way with plural and genitive "suffixes."

2. Automata

ENGLISH AUTOMATA contains six finite state transducers that take care of English spelling conventions. The first of these is a trivial one (it maps every character to itself -- needed for technical reasons); the remaining five encode spelling rules. These rules assume that suffixes are specified as follows:

```
+s      - plural in nouns and 3rd singular in verbs
+ed     - past tense and past participle
+ing    - progressive
+er     - comparative
+est    - superlative
+er     - verb-to-noun agent marker
+ly     - adverb marker
's      - singular genitive
'       - plural genitive
+able   - verb-to-adjective "able" marker
```

The automata correspond to the following rules.[2]

Epenthesis: +/e <--> { {c | s (h) } | S | y/i } __ s

Lexical + corresponds to surface e after ch, sh, s, x, z, or y/i before s; otherwise, lexical + corresponds to 0, the empty string.

Examples: foxes, churches, spies (+.e)
cats, skis, boys (+.0)

This treatment of spelling conventions for the plural and third person singular suffixes assumes that the suffix consists of

---

[1] See elsewhere in this issue for an explanation of lexical entries.

[2] To avoid confusion between parentheses indicating optionality and parentheses indicating a lexical-surface correspondence pair, we have used a slash notation to indicate correspondence pairs in the rules themselves. Thus y/i, for example, is equivalent to (y.i).

s only and that the e is inserted in certain environments. We make no claims that this analysis is in any way preferable to the alternative of treating the suffix as es with deletion taking place in the complementary environments.

Gemination: +/<Cl> <--> {` | #} C* V <Cl> __ V
where Cl = {b, d, f, g, l, m, n, p, r, s, t}
= word boundary[3]

Lexical + is realized as a copy of a preceding b,d,...,t when the latter follows a single vowel and the vowel is stressed; as 0 elsewhere.

Examples: referring, bigger (gemination)
traveling, cooler (no gemination)

Stress proves to be important for this English description only for the purpose of spelling conventions for gemination. A single quotation mark (the left, or opening, quotation) indicates that the following syllable is stressed. The system considers the first syllable stressed unless it encounters a subsequent stress marker, so it is not necessary to mark stress for single-syllable words or for words stressed on the first syllable.[4]

Y-replacement: y/i <--> C __ +/= - {i, a}

After a consonant, lexical y corresponds to i when a lexical suffix marker and any pair other than i/i or a/a follows; to y elsewhere.

Examples: rallies, spies, spied, happily (y.i)
days, spying, played, carryable (y.y)

The right context for this rule, (+.=) -{i, a}, in effect specifies that the string must be either (+.e) (s.s) or (+.0){e,

---

[3]The word boundary symbol is not a part of Kimmo's proposed formalism. It is used here simply for convenience. The automaton achieves the result through a state that is reached only after a vowel is seen and the left context for gemination not satisfied. To exit from this state, a stress mark must be subsequently encountered.

[4]Exceptions that meet the SD of the rule but are not geminated, e.g., com`bative, can be handled in the current system simply by not marking stress in the stem. A stress mark can be artificially inserted for those words that do not have stress on the stem-final syllable but are nevertheless geminated, e.g., worshipped.

1}. This effect is a product not only of this particular rule, but also of the makeup of the English dictionary as well as the operation of other automata. We will not attempt to detail all the interrelationships--interested readers may note interactions with the two rules above as well as with the possible suffixes in the English dictionary--nevertheless, the relative nature of Kimmo's rules should be clear.

Elision: e/0 <--> C2 __ +/0 V
-V V __ +/0 e
{g | c} __ +/0 { e | i }
where C2 = {CP | CP IN -V & CP IN -{c, g}}

Lexical e corresponds to surface 0 in the following contexts:

(1) After any non-vowel pair except c or g and before a suffix beginning with a vowel.

(2) After a single vowel and before a suffix begining with e.

(3) After c or g and before a suffix beginning with e or i.

Otherwise, e corresponds to e.

Examples: movable, agreed, hoed, larger, moving, racing (e.0)
agreeing, agrees, hoeing, moves, raceable (e.e)

This rule, made complex because of the three sets of environments, points out a possible area for revision in the formalism. The formalism currently leaves no choice but to collapse all environments affecting a single correspondence pair into a single rule to achieve context restriction.[5] This condition is unmotivated from a linguistic point of view; there seems to be no principled reason for preventing two independent rules from referring to the same correspondence pair. At present it would be possible to write three separate surface coercion rules (indicated by the symbol <--) with the effect of forcing lexical e to be realized as surface 0 in these environments. But we would still need to put the three environments together into one rule to force lexical e be realized as surface e everywhere else. This could be

[5]See Karttunen's other paper in this issue for comment.

avoided by adopting a general convention that any ordinary lexical character (as opposed to diacritics, boundary symbols, etc.) has to be realized as the same character on the surface unless some rule permits otherwise.

I-replacement: i/y <--> __ e/0 +/0 i

Lexical i corresponds to surface y if followed by e/0 +/0 i/i; otherwise to surface i.

Examples: dying, lying (i.y)
died, lied (i.i)

These rules take care of most regular morphological variation in English but obviously they do not cover everything. For example, alternate spellings such as the ey+s/ie+s in trolley+s/trollies have been left out. Also, some comparatives of adjectives ending in y, e.g. shy+er/shyer (*shier), are not treated correctly. These present no problem but they would add a bit more clutter.[6]

3. Prefixes

Prefixes present some interesting problems for Kimmo's system since the continuation class for prefixes must consist of some subset of the Root lexicon. Our English description currently overgenerates, accepting un followed by any root in the lexicon.

The prefix un is of particular interest since actually there are two uns in English. One combines with certain adjectives and is fully productive with past participles of transitive verbs; it has the meaning "not," as in unmoved. The second combines only with a rather limited class of verbs and has the approximate paraphrase "the reverse of (verb)," as in uncover. These facts account for the ambiguity of forms such as uncovered, which is most naturally taken in the first sense in an uncovered head and in the second sense in the plot was uncovered.

Thus we would like a morphological analyzer to return two analyses for uncovered, but only one for unmoved (un = not) or for uncover (un = the reverse of). The present description

---

[6] A natural solution for shy in this framework would be to use a special character, e.g. Y, in the lexical representation whose only surface realization is y. "Y-Replacement" would then not be triggered.

accomplishes this but at the cost of extra listings for each root that combines with un (reverse of) and an overgeneration of un (not) forms. There are at least three options for improving this aspect of an English description within Kimmo's general framework.

First, it would be possible to use two different pairs of diacritics for the two uns. Two transducers then could check to make sure that when the first member of either of the pairs is encountered, the second member is encountered also. Those verb roots in the lexicon which can combine with un (the reverse of) could each be marked with the appropriate diacritic, and since participles generally can occur with the other un, the second member of its diacritic pair could appear in the participle lexicons.

Although we have tested this approach and it gives the desired results, it is perhaps an artificial means of accounting for what may be more general organizing principles in the lexicon.[7] Another disadvantage might be the computational burden of proliferating transducers if diacritics were used heavily in the morphological description. Also, one has the undesirable side-effect of a lexicon which returns strings that include not just letters, but an arbitrary number of diacritics as well.

Second, one could restrict the continuation classes by separating the root lexicon into many sublexicons. Then the continuation classes for un could be defined as the appropriate sublexicon. A disadvantage of this approach is that much of the lexical information would have to be duplicated. English has many homophonous noun-verb pairs and creating separate lextrees for nouns and verbs would necessitate the duplication of as many memory cells as there are letters in the total number of English homophones in the lexicon. A second problem is that the continuation class for un (not) does not actually consist of verbs per se, but of past participles. The morphosyntactic implementation should be able in this case to have labeling properties in order to refer to derived structure. A separate lextree for past participles is surely impractical.

A third possibility, requiring a substantial revision of Kimmo's system, is to describe the morphosyntax using phrase structure rules augmented with features. Just as a feature

---

[7] See also the comments on this approach in the morphosyntax section of Karttunen's paper in this issue.

matching procedure would be used for subject-verb agreement, it could be used to ensure that prefixes would combine with the correct forms. Kimmo's system already has the facility for including features with each item in the lexicon. It would be relatively simple to include a process that would accept only those complex strings which passed a feature-matching test. We expect that future versions of Two-level morphology will incorporate such a revision of the morphosyntactic component.

4. Conclusion

The analysis we have presented here should be viewed as no more than an exploration of Two-level morphology as a description of English. We may have relied at times on convenience rather than principle in some of these rules and treatments. With further research, however, restrictions on rule forms and the form of the lexicon will no doubt be proposed. Apart from the particulars of these rules, we believe that the analysis of English, together with other analyses in this issue, demonstrates that Two-level morphology is powerful enough to handle most low-level morphological variation. Other aspects related to the morphosyntax, however, were more of a challenge to Kimmo's system and suggest that revisions are necessary.

A. English Dictionary

```
ALTERNATIONS
     ( /N = N )
     ( /IN = C1 )
     ( /MN = MN )
     ( /V = P3 PS PP PR I AG AB )
     ( /IV1 = PR I AG AB )
     ( /IV2 = P3 PR I AG AB )
     ( /A = PA CA CS Ly )
     ( /IP3 = IP3 )
     ( /IPS = IPS )
     ( /IPP = IPP )
     ( C1 = C1 )
     ( C2 = C2 )
     ( I = I )
     ( P3 = P3 )
     ( PS = PS )
     ( PP = PP )
     ( PR = PR )
     ( AG = AG )
     ( PA = PA )
     ( CA = CA )
     ( CS = CS )
     ( Ly = Ly )
     ( Root = Root )
     ( AB = AB )
     ( # = )
END

LEXICON N       0       C1  "N SG";       +s   C2  "N PL"
LEXICON MN      0       C2  "MASS N"
LEXICON C1      0       #   "";           's   #   " GEN"
LEXICON C2      0       #   "";           '    #   " GEN"
LEXICON P3      +s      #   "V PRES SG 3RD"
LEXICON IP3     0       #   "V PRES SG 3RD"
LEXICON PS      +ed     #   "V PAST"
LEXICON IPS     0       #   "V PAST"
LEXICON PP      +ed     #   "V PAST PRT"
LEXICON IPP     0       #   "V PAST PRT"
LEXICON IP      0       #   "V PAST"
LEXICON PR      +ing    #   "V PROG"
LEXICON I       0       #   "V"
LEXICON IP1     0       #   "V PRES SING 1ST"
LEXICON AG      +er     /N  "AG "
LEXICON PA      0       #   "A"
LEXICON CA      +er     #   "A COMP"
LEXICON CS      +est    #   "A SUP"
LEXICON Ly      ly      #   "ADV"
LEXICON AB      +able   #   "VERB ABL"
```

```
LEXICON Root
     am          #       "V PRES SING 1ST";
     am          #       "AUX";
     are         #       "AUX";
     are         #       "V PRES PL";
     are         #       "V PRES SING 2ND";
     at          #       "PREP";
     at`tack     /N      "";
     at`tack     /V      "";
     a`gree      /V      "";
     be          #       "AUX";
     be          /IV1    "";
     beer        /N      "";
     believe     /V      "";
     big         /A      "";
     bit         /IPS    "";
     bite        /IV2    "";
     bitten      /IPP    "";
     boo         /V      "";
     boy         /N      "";
     cacti       /IN     "N PL";
     cactus      /IN     "N SG";
     cat         /N      "";
     church      /N      "";
     cool        /A      "";
     day         /N      "";
     did         /IPS    "";
     die         /V      "";
     do          /IV1    "";
     does        /IP3    "";
     done        /IPP    "";
     fox         /N      "";
     go          /IV1    "";
     goes        /IP3    "";
     gone        /IPP    "";
     grouch      /N      "";
     had         /IPS    "";
     had         #       "AUX";
     has         /IP3    "";
     has         #       "AUX";
     have        #       "AUX";
     have        /IV1    "";
     ice         /MN     "";
     industry    /N      "";
     is          /IP3    "";
     kill        /V      "";
     kiss        /N      "";
     kiss        /V      "";
     mice        /IN     "N PL";
     milk        /MN     "";
     mouse       /IN     "N SG";
     oc`cur      /V      "";
     race        /N      "";
```

```
    race          /V       "";
    rally         /N       "";
    refer`ee      /N       "";
    refer`ee      /V       "";
    re`fer        /V       "";
    ski           /N       "";
    sleep         /IV2     "";
    slept         /IPP     "";
    slept         /IPS     "";
    spy           /N       "";
    spy           /V       "";
    tie           /V       "";
    tiptoe        /V       "";
    toe           /N       "";
    travel        /N       "";
    travel        /V       "";
    try           /V       "";
    un            Root     "NEG";
    under`stand   /IV2      "";
    under`stood   /IPP      "";
    under`stood   /IPS      "";
    undid         /IPS     "";
    undo          /IV1     "";
    undoes        /IP3     "";
    undone        /IPP     "";
    untie         /V       "";
    went          /IPS     "";
    went          /IPS     ""
END
```

B. English Automata

```
ALPHABET
     a b c d e f g h i j k l m n o p q r s t u v w x y z '
     + `
     NULL 0
     ANY  =
     SUBSET V a e i o u
     SUBSET C b c d f g h j k l m n p q r s t v w x z
     SUBSET S s x z
     END

"Surface Characters "   1  28
      a b c d e f g h i j k l m n o p q r s t u v w x y z ' =
      a b c d e f g h i j k l m n o p q r s t u v w x y z ' =
 1:   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

"Epenthesis"              6 8
      c h s S y + + =
      c h s S i e 0 =
 1:   2 1 4 3 3 0 1 1
 2:   2 3 3 3 3 0 1 1
 3:   2 1 3 3 3 5 6 1
 4:   2 3 3 3 3 5 6 1
 5.   0 0 1 0 0 0 0 0
 6:   1 1 0 1 1 1 1 1

"Gemination"                                                    16 26
     V  b  d  f  g  l  m  n  p  r  s  t + + + + + + + + + + + + ` =
     V  b  d  f  g  l  m  n  p  r  s  t  b  d  f  g  l  m  n  p  r  s  t 0 0 =
 1:  4  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  1 1 1
 2.  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 1 0
 3:  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1 1 0
 4: 16  5  6  7  8  9 10 11 12 13 14 15  0  0  0  0  0  0  0  0  0  0  1 116
 5: 16 16 16 16 16 16 16 16 16 16 16 16  2  0  0  0  0  0  0  0  0  0  3 1 1
 6: 16 16 16 16 16 16 16 16 16 16 16 16  0  2  0  0  0  0  0  0  0  0  3 1 1
 7: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  2  0  0  0  0  0  0  0  3 1 1
 8: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  0  2  0  0  0  0  0  0  3 1 1
 9: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  0  0  2  0  0  0  0  0  3 1 1
10: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  0  0  0  2  0  0  0  0  3 1 1
11: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  0  0  0  0  2  0  0  0  3 1 1
12: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  0  0  0  0  0  2  0  0  3 1 1
13: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  0  0  0  0  0  0  2  0  3 1 1
14: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  0  0  0  0  0  0  0  2  3 1 1
15: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  0  0  0  0  0  0  0  0  2 3 1 1
16: 16 16 16 16 16 16 16 16 16 16 16 16  0  0  0  0  0  0  0  0  0  0 016 116
```

```
"Y-Spelling"          6 7
      C y y + i a =
      C y i = i a =
 1:   2 1 0 1 1 1 1
 2:   2 5 3 1 1 1 1
 3.   0 0 0 4 0 0 0
 4.   1 1 0 1 0 0 1
 5:   1 1 0 6 1 1 1
 6.   0 0 0 0 1 1 0


"Elision"      15 8
      V i  e  e  +  g  c  =
      V i  e  0  0  g  c  =
 1:   2 2  3  4  1 11 11  1
 2:   1 1  5  6  1 11 11  1
 3:   1 1  5  6  9 11 11  1
 4.   0 0  0  0  8  0  0  0
 5:   1 1  1  4  7 11 11  1
 6.   0 0  0  0 10  0  0  0
 7.   1 1  0  0  0  1  1  1
 8.   1 1  1  0  0  0  0  0
 9.   0 0  0  0  0  1  1  1
10.   0 0  1  0  0  0  0  0
11:   1 1 14 12  1  1  1  1
12.   0 0  0  0 13  0  0  0
13.   0 1  1  0  0  0  0  0
14:   1 1  1  0 15 11 11  1
15:   1 0  0  0  1 11 11  1


"I-Spelling"       7 6
      i e + i e =
      y 0 0 i e =
1:    2 1 1 5 1 1
2.    0 3 0 0 0 0
3.    0 0 4 0 0 0
4.    0 0 0 1 0 0
5:    1 1 1 1 6 1
6:    0 1 7 0 0 1
7.    0 0 0 0 0 1

END
```