Morphology with Two-Level Rules and Negative Rule Features

John Bear
Artificial Intelligence Center and
Center for the Study of Language and Information
SRI International

Abstract

Two-level phonology, as currently practiced, has two severe limitations. One is that phonological generalizations are generally expressed in terms of transition tables of finite-state automata, and these tables are cumbersome to develop and refine. The other is that lexical idiosyncrasy is encoded by introducing arbitrary diacritics into the spelling of a morpheme. This paper explains how phonological rules may be employed instead of transition tables and describes a more elegant way of expressing phonological irregularity than with arbitrary diacritics, making use of the fact that generalizations are expressed with rules instead of automata.

1 Introduction

The theme of this paper is how to deal with the phonological or orthographic half of the problem of computational morphology, i.e., how to handle the various problems associated with the spellings of morphemes. The examples in this paper have been drawn from English orthography but it is easy to find examples from other languages where these techniques would be applicable as well.

In an earlier paper [2], I presented a formalism for two-level phonological (or orthographic) rules very similar to Koskenneimi's [8] and described how rules in that formalism could be interpreted in a computational system. There were problems with both my formalism and Koskenniemi's that could have been solved with the device of negative rule features. In this paper I discuss these problems and their solutions.

2 Historical Note

The formalism described here was developed with the goal of allowing the linguist to write rules with similar or even identical contexts and still have a way of processing them. This stands in contrast to Koskenniemi's formalism, which, in its initial formulation, seemed to rule out pairs of such rules.

For instance, in Koskenniemi's formalism, as originally stated, the two rules below,

$a : b \Longleftarrow \alpha\_\_\beta$
$a : c \Longleftarrow \alpha\_\_\beta,$

would clash. Together they assert that a lexical character /a/, preceded by a sequence of character pairs α and followed by a sequence of character pairs β, must correspond to *both* /b/ and /c/ on the surface.

The orthographic rules described here are used in a morphological analysis system that is based on the work of Koskenniemi, Karttunen, and Wittenburg [8,5]. Its morphosyntactic component uses, instead of continuation classes, an extension of PATR type rules including a device described by Karttunen [4] for handling disjunction. One version of this system also uses a definite-clause grammar in addition to the PATR-type unification, and disjunction. It has been implemented in Prolog and runs on a Sun.

3 Summary of Alternative Rule Formalism

The basic idea behind the notion of two-level rule (due to Koskenniemi [8]) is that there are two levels of linguistic information to which a rule may refer. One has to do with how a morpheme is spelled in the lexicon. That is called the lexical level. The other has to do with how a morpheme appears in text, i.e., the surface representation. There is no way for rules to apply one after the other, creating and referring to intermediate levels of representation. Instead, rules are viewed as constraints on mappings between surface and underlying forms of morphemes. They stipulate how to get from underlying to surface form, and vice versa.

Two-level rules in the alternative to Koskenniemi's formalism that I proposed in an earlier paper [2], take one of three forms:

1) $a \longrightarrow b/\alpha\_\_\beta$
2) a/b *allowed* $/\alpha\_\_\beta$
3) a/b *disallowed* $/\alpha\_\_\beta$

The α and β in the contexts of these rules represent strings of character *pairs* where one character of the pair refers to the lexical level of representation and the other refers to the surface.

Rule (1) is very similar to a standard phonological rule. It means roughly that lexical /a/ must correspond to surface /b/ in the context given. A more accurate and detailed description is as follows: if lexical /a/ occurs in the given context, then it may not correspond to what it normally would correspond to, but it may correspond to surface /b/.

Rule (2) means that lexical /a/ is allowed to correspond to surface /b/ in the context given, but not elsewhere. More precisely, the rule allows the pair /a:b/ (lexical /a/ corresponding to surface /b/) to occur in the context given and, unless there are other rules licensing the pair in other contexts, the context given is the only place where that correspondence is allowed.

Rule (3) says that lexical /a/ may not correspond to surface /b/ in the context given. Both rules (1) and (2) mention a character's default. A normal alphabetic character in this system defaults to itself. This means that a pair of alphabetic characters /a:a/ does not need to be licensed by a rule. In contrast to alphabetic characters (a through z), there are diacritic characters such as the plus sign (+) for morpheme boundaries. In Karttunen and Wittenburg's system, [5] there is also a backquote (‘) for representing stress; Koskenniemi uses several others as well, [8]. The default for lexical-level diacritics, at least in the

system described here, is that they correspond to the null surface character, which is frequently written with a zero.

4 Negative Rule Features

There is a problem with previous accounts of English that have been done in terms of two-level rules. There is no easy way to let the phonological rules know about individual idiosyncrasy in the lexical items. In the work of Koskenniemi [8] and Karttunen and Wittenburg [5], diacritics are put into the lexical representation of a word in order to allow the linguist to write a phonological rule that applies in some words and not others according to the presence or absence of the diacritic. The diacritic is mentioned in the rule. The words that do not contain the diacritic do not undergo the rule.

In old-fashioned generative phonology, there was the notion of a negative rule feature to handle such cases. One could say of certain morphemes that appeared to be exceptions to certain phonological rules that such morphemes possessed a feature specifying that some particular phonological rule did not apply to them[1].

The idea of negative rule features has an advantage over the use of diacritics mentioned above in that it allows simplification of the phonological rules and the lexicon. It seems to me more straightforward to have a lexical item that says minus such and such a rule than to have the lexical item contain a colon or quotation mark whose function is to assert that some rule does not apply. The complexity of the lexical items is the same, but in the first case, at least, the phonological rule can be made simpler by omission of the arbitrary diacritic.

There are three examples from English orthography that will be used to help demonstrate how negative rule features may be employed.

The analysis of consonant gemination in Karttunen and Wittenburg's paper, [5], relies on the use of diacritics of just the sort mentioned above. A simplified version of the rule is given below.

> Gemination:
> $+ : c1 \Longleftrightarrow$ ' $C * V$ =: $c1$ _V;
> where c1 is in
> {b,d,f,g,l,m,n,p,r,s,t}.

This rule uses a plus sign (+) for morpheme boundaries, and a backquote (') for accent where accent is important. It correctly describes the following data:

> questioning versus *questionning,
> debiting versus *debitting,
> eating versus *eatting.

The rule also correctly describes the following data, provided the lexical entry contains a backquote in the right place.

> referred versus. *refered (spellings in lexicon are "re'fer" + "ed").

In order to get the facts right for monosyllabic words, Karttunen and Wittenburg's rule also mentions that, instead of a backquote, a word boundary (#) will do.

The only point of contention here is that their system requires the the lexical entry to contain a diacritic (and furthermore the diacritic must be correctly located within the word). That the diacritic is reminiscent of an accent mark is no accident. Stress is clearly a factor in English consonant gemination. Their solution is to find a way to represent stress in the orthography. The alternative proposed here is to express it in the form of a negative rule feature on the following sample lexical items. The rule is again simplified.

[1]For instance, see Schane [11], pp. 108-109

> Gemination rule:
> $+ \longrightarrow c1 / C\ V\ c1$ _V;
> where c1 is in {b,d,f,g,l,m,n,p,r,s,t}
> Words:
> refer (default is that it is consistent with all rules)
> bother
> -gemination (means that the gemination rule does not apply to this word)

There are other sets of data for which this technique is useful. The case that comes to mind most readily deals with combining a noun or verb stem ending in /o/ with an /s/ morpheme representing, respectively, plural for nouns and third person singular for verbs. The following rules do well at describing these facts about English orthography.

> EPENTHESIS RULES:
> epenthesis1:
> $+ \longrightarrow e$ / o _ s.
> epenthesis2:
> $+/e$ *allowed in context* o _ s.
>
> DATA:
> potato+s ⟹ potatoes, *potatos (need an /e/)
> do+s ⟹ does, *dos (need an /e/)
> piano+s ⟹ pianos, *pianoes (can't have an /e/)
> piccolo+s ⟹ piccolos, *piccoloes (can't have an /e/)
> banjo+s ⟹ banjos or banjoes (both are acceptable)
> cargo+s ⟹ cargos or cargoes (both are acceptable)

The first of the epenthesis rules describes /potato+s/ ⟹ [potatoes] and /do+s/ ⟹ [does] correctly, but incorrectly states that the plural of /piano/ is */pianoes/. The second rule is weaker, generating all of the correct forms – but all of the wrong ones too, so that it achieves the right results for /banjo+s/ ⟹ [banjoes] or [banjos] and likewise for /cargo+s/, but yields both the right and the wrong results for the others.

The way to get the facts right is to put negative rule features on the lexical items in question, as shown here:

> LEXICON
> piano
> - [epenthesis1 epenthesis2]
> piccolo
> - [epenthesis1 epenthesis2]
> banjo
> - epenthesis1
> cargo
> - epenthesis1
> potato
> do

The alternatives are either to list some forms as being irregular or to insert diacritics into some of the words so that the rule(s) will apply only to the correct lexical items. To list some of the forms as irregular is to miss the generalization that they are all irregular in exactly the same way. To use a diacritic (or possibly two) to describe the facts correctly may lead to making other, unrelated rules more complicated. Furthermore, it seems to be an attempt at expressing historical information, such as a word's provenance, in terms of abstract phonological segments.

In general, the device of negative rule features seems to be well suited to the task of passing information between a lexical entry and the phonology component. This is a useful capability. It is perhaps analogous to employing augmented phrase-structure rules in syntax when, at least in theory, pure context-free rules would do.

The main idea here is that there is a way to let phonological (or orthographic) rules refer to features of a morpheme that may

not be easily represented as phonemic segements. As regards the gemination rule mentioned earlier, the right procedure might be to let the rule mention stress and store values for that feature in the lexical entries.

5 Computer Interpretation of the Rules

What makes these rules interesting is that there is a way to apply them in a morphological parser or generator. What follows is a description of the algorithm used by the code that I have implemented in Quintus Prolog on a Sun. When the rule epenthesis1 is read in, it is decomposed into two rules. This rule,

epenthesis1:
$+ \longrightarrow e \ / \ o \_\_ s,$

yields these rules:

epenthesis1:
$+/e$ *allowed in context* $o \_\_ s$
epenthesis1:
$+/0$ *disallowed in context* $o \_\_ s$.

These rules are then stored as lists of character pairs:

epenthesis1:
allowed: $o/o \ . \ +/e \ s/s$
epenthesis1:
disallowed: $o/o \ +/0 \ s/s$.

6 Basic Algorithm

The rules are sequences of character pairs. A mapping between a string of lexical characters and a string of surface characters may also be considered to be a list of character pairs. No disallowed-type rule may be a substring of a mapping between a lexical string and a surface string.

The rule checker proceeds down the list of character pairs, looking for any substring that is the same as one of the disallowed-type rules. If it finds one, the string of character pairs it was considering is not a valid mapping from a lexical form (word) to a surface form.

The other type of rule, the allowed-type rule, is somewhat different. A dot is put into the rule right after the end of the left context to mark the next character pair as being the main pair of the rule. Any character pair that is the main pair for one of these allowed-type rules needs to be surrounded by the right and left contexts of one of these rules. The way that is checked for in this system is as follows. The string of charcter pairs is scanned from left to right. Each time a pair is encountered that is the same as the first pair of some allowed-type rules, the rules are put into a set. As more character pairs are scanned, they are compared with the sets of rules already encountered. Rules that do not continue to match the scanned input are ejected from the set. When the main pair of a rule in one of these sets is scanned, it is removed from the set it was in and put into a new one. The rules in this set are compared with scanned input in the same manner as before except that, if the last pair of some rule matches a pair that is being scanned, the whole set is discarded as no longer of interest. Conversely, if there is not at least one rule in the set that matches the scanned input all the way to the end, then the input being scanned is not an allowable mapping between lexical and surface forms.

7 Algorithm With Negative Rule Features

Thus far, nothing has been said about how negative rule features enter into the picture. When a morpheme boundary is encountered, a morpheme has just been looked up in the lexicon. At that point, if it has some negative rule features on it, it is a simple matter to sort through the list of rules that have partially matched the input and discard those that the morpheme says do not apply. If that entails eliminating the last rule in some set of allowed-type rules that have all already matched past the main pair of the rule, then the input being scanned is not allowable as a possible mapping between lexical and surface forms. Otherwise one should just go on as before, comparing the rules with the input being scanned.

8 Conclusion

A general procedure for using phonological or orthographic two-level rules has been presented. These rules are much easier to refine and develop than automata transition tables. In addition, a method has been presented for listing which morphemes are exceptions to which [orthographic] rules, and an algorithm has been described that makes it possible to use this information in a straightforward way.

Furthermore, these are two-level rules. As Koskenniemi has noted, [8], since these rules simply state correspondences between surface strings and underlying strings, they may be used either for doing generation or recognition. The device of negative rule features proposed here has the same power as Koskeniemi's device of putting arbitrary diacritics into selected classes of morphemes and rules, but is argued to be simpler.

Acknowledgments

I would like to thank Meg Withgott for helpful comments on this topic. I have also benefited greatly from conversations with Lauri Karttunen and Kimmo Koskenniemi regarding the general problem of two-level phonology. This research was funded by the Defense Advanced Research Projects Agency under Office of Naval Research Contract N00014-85-C-0013.

References

[1] Bear, John (1985) "Interpreting Two-level Rules Directly," presented at a Stanford workshop on finite-state morphology.

[2] Bear, John (1986) "A Morphological Recognizer with Syntactic and Phonological Rules," *COLING 86.*

[3] Karttunen, Lauri (1983) "Kimmo: A General Morphological Processor," in *Texas Linguistic Forum #22*, Dalrymple et al., eds., Linguistics Department, University of Texas, Austin, Texas.

[4] Karttunen, Lauri (1984) "Features and Values," in *COLING 84.*

[5] Karttunen, Lauri and Kent Wittenburg (1983) "A Two-level Morphological Analysis Of English," in *Texas Linguistic Forum #22*, Dalrymple et al., eds., Linguistics Department, University of Texas, Austin, Texas.

[6] Kay, Martin (1983) "When Meta-rules are not Meta-rules," in K. Sparck-Jones, and Y. Wilks, eds. *Automatic Natural Language Processing*, John Wiley and Sons, New York, New York.

[7] Kay, Martin (1987) "Nonconcatenative Finite-State Morphology," paper presented at a workshop on Arabic Morphology, Stanford University, Stanford, California.

[8] Koskenniemi, Kimmo (1983) *Two-level Morphology: A General Computational Model for Word-form Recognition and Production.* Publication No. 11 of the University of Helsinki Department of General Linguistics, Helsinki, Finland.

[9] Koskenniemi, Kimmo (1983) "Two-level Model for Morphological Analysis," *IJCAI 83*, pp. 683-685.

[10] Koskenniemi, Kimmo (1984) "A General Computational Model for Word-form Recognition and Production," *COLING 84*, pp. 178-181.

[11] Schane, Sanford (1973) *Generative Phonology*, Prentice Hall, Englewood Cliffs, New Jersey.

[12] Selkirk, Elizabeth (1982) *The Syntax of Words*, MIT Press, Cambridge, Massachussetts.

[13] Shieber, Stuart (1986) *An Introduction to Unification-Based Approaches to Grammar*, CSLI Lecture Notes Series, Stanford University, Stanford, California.