PHONOLOGICAL ANALYSIS AND OPAQUE RULE ORDERS

Michael Maxwell
Summer Institute of Linguistics
Box 248
Waxhaw, NC 28173 USA

ABSTRACT

General morphological/ phonological analysis using ordered phonological rules has appeared to be computationally expensive, because ambiguities in feature values arising when phonological rules are "un-applied" multiply with additional rules. But in fact those ambiguities can be largely ignored until lexical lookup, since the underlying values of altered features are needed only in the case of rare opaque rule orderings, and not always then.

INTRODUCTION

While syntactic parsing has a long and illustrious history, comparatively little work has been done on general morphological and phonological parsing — what I will call, for lack of a better term, "morphing."

The morphological and phonological parsing programs which do exist are, for the most part, either restricted to a single language or, like FONOL (Brandon 1988), are limited to generating surface forms from underlying forms. Two exceptions to this generalization are Kimmo (see Koskenniemi 1984, and the papers in *Texas Linguistic Forum* 22) and AMPLE (Weber, Black and McConnel 1988). However, Kimmo implements a non-standard theory of phonology, while AMPLE implements an item-and-arrangement morpher with virtually no allowance for (morpho-)phonological rules.

One reason for the paucity of general morphing programs is the apparent computational complexity of morphing. Phonological rules of natural language include deletion rules, which means that they potentially represent an unrestricted rewriting system. But in fact people routinely parse words into their constituent morphemes, which implies that Universal Grammar must place strong restrictions on phonology and morphology, effectively reducing the complexity of morphing. To the extent that linguists can analyze such restrictions, we may be able to reduce the computational complexity of morphing.[1] This paper investigates how one such restriction, a restriction on interaction among multiple rules, can be taken advantage of.

While linguists treat phonological rules as rules which derive surface forms from underlying forms, a program analyzing the surface strings of a language must "un-apply" those rules to a surface form to discover its underlying form. Most phonological rules have a neutralizing effect when applied in the derivational (synthesis) direction; accordingly, when a rule is un-applied, there will in general be more than one way to undo its effects. In a computational setting, this implies the need to restrict the search space, lest those ambiguities multiply with the application of multiple rules. This paper discusses a way of restricting that search space.

ASSUMPTIONS

For purposes of discussion, I will consider a morpher which implements a morphophonological theory of the following type. Phonological rules are written in the "standard" way with distinctive features but without any abbreviatory conventions (parentheses, curly braces, angled brackets, alpha variables, etc.); the rules apply in linear order, the output of each serving as the input to the next. I will assume that distinctive features are binary, although the results will apply in an analogous way to (finitely) multiply-valued features. For the most part, I will ignore the multiple application problem. I will not explicitly discuss morphological rules, but we may assume they apply either in a block (pre-cyclically) or cyclically. The resulting system resembles that of *The Sound Pattern of English* (Chomsky and Halle 1968, henceforth *SPE),* but without the abbreviatory schemata.

---

[1]Even without being able to explicitly state the restrictions, it may be that a correctly formulated set of rules for a given language will turn out to be readily parsable. However, that hope relies on the linguist to properly formulate the rules. I will return to this point later.

From a computational perspective, the working cycle of the morpher is as follows: phonological rules are un-applied in linear order to a form (assumed to be in an unambiguous phonetic representation), and then one or more morphological rules are un-applied (one in the case of cyclic rule ordering, one or more with non-cyclic rule ordering). By un-application of a rule, I mean applying it in reverse: going from a (more) surface form to a (more) underlying form. Lexical lookup is attempted after each morphological rule is un-applied. Lexical lookup acts then as a filter; if lexical lookup is successful, the set of phonological and morphological rules which were un-applied represents a successful derivation (modulo certain later tests, not discussed here), otherwise not. This is the classical approach to computational morphology/ phonology, as described in Kay (1977).[2]

THE PROBLEM

The problem to be explored in this paper arises when un-application of a rule results in one or more ambiguous feature values. Consider, as a simple case, the following rule:

$$\begin{bmatrix} +\text{syllabic} \\ -\text{cons} \\ -\text{stress} \end{bmatrix} \longrightarrow \emptyset \,/\, \begin{bmatrix} C \\ -\text{vd} \end{bmatrix} \text{——} \begin{bmatrix} C \\ -\text{vd} \end{bmatrix}$$

Suppose this rule is used to analyze a word which, on the surface, has two adjacent voiceless consonants. The rule specifies only three features for the vowel to be epenthesized in analysis of the surface form (i.e. the vowel which was deleted to generate the surface form): [+syllabic -cons -stress]. The remaining features must be "guessed" during analysis. Since this involves multiple features, the combinatorial possibilities are many. In addition, there is the possibility that no vowel should be epenthesized — that the consonants were adjacent underlyingly.

---

[2]The morpher discussed in the text is being implemented as one module of the planned "Hermit Crab" system (a syntactic parser and possibly a functional structure module being additional modules). Hermit Crab takes its name from the fact that the internal rule system (the "crab") has a rule structure which will, in general, depart from the rule structure as viewed by the user (who sees only the "shell").

This problem is not limited to rules of deletion. Any rule which neutralizes an underlying contrast will cause ambiguity (albeit not usually as great as in the case of deletion rules) when the rule is un-applied.

The difficulty is compounded by the interaction of multiple rules. Anderson (1988) suggests a "typical" rule depth in natural languages of 15-20 rules. Clearly the possibilities of computational explosion loom large.

The remainder of this paper will investigate some approaches to this problem.

THE BRUTE FORCE APPROACH

I will first explore the following brute-force technique: when a phonological rule is un-applied, instantiate all possible combinations of features changed by the rule onto the new form output by the rule.

For a deletion rule, the number of feature combinations which may be instantiated is 2^n, where n = the number of features not specified in the left-hand side of the rule. For concreteness, consider the vowel deletion rule discussed above. In the SPE system I count eighteen distinctive features (not including certain prosodic features). Subtracting the three features whose values are supplied by the rule leaves fifteen unspecified features. Since 2^{15} is a very large number, there is clearly a need for pruning the search space.

A certain amount of pruning comes readily. One can begin by eliminating universally impossible feature cooccurences. For instance, if a segment is [+syllabic], it cannot be [-continuant]. In the case of the vowel deletion rule, this reduces the search space to about 2^8 combinations. (The eight features in the SPE system whose values are not determined by the [+syll -cons -stress] features of the rule are: High, Low, Back, Round, Tense, Voiced, Covered and Nasal. Some combinations of these are also mutually incompatible, e.g. [+high +low], reducing the search space slightly more.) We can do still better by eliminating noncontrastive features in the language we are working with. For Spanish, for instance, we could eliminate the features *Covered* and *Nasal* if we work with the surface vowels (ignoring the light nasalization of vowels before nasal consonants), and the features *Tense* and *Voiced* if we limit ourselves to features appearing only in

underlying vowels. (The assumption here is that tensing and voicing, which in most dialects of Spanish are predictable, do not condition other rules.) These reductions leave a search space of $2^4 = 16$. We can limit this still further by eliminating combinations of features which do not occur in a particular language ([-back +round], for instance). We are left with an irreducible search space of five combinations of features in this case — the five vowels which occur (underlyingly) in standard Spanish. This last reduction constitutes the use of Segment Structure Conditions (SSCs) to constrain rule un-application. This may be done rapidly by consulting a list of possible segments of the language. (The list of possible segments need not be confined to those appearing at the surface; i.e. absolute neutralization can be accommodated by allowing for absolutely neutralized segments in the SSCs.)

Since this rule is a deletion rule, we must also allow for the situation in which no vowel was deleted, increasing the search space by one.

Similarly, non-deletion rules introduce an ambiguity of 2^m, where m = the number of features on the right-hand side of the rule, often pruneable by reference to SSCs.[3] Consider, for example, a language in which the only coronal obstruants are *t* and *č*, and the following rule:

$$\begin{bmatrix} +\text{cor} \\ -\text{cont} \end{bmatrix} \longrightarrow \begin{bmatrix} -\text{ant} \\ +\text{del rel} \end{bmatrix} \; / \; \_\_ \text{i}$$

Naive un-application of this rule to the sequence *či* would lead to a four-way ambiguity in the values of the feature set {ant, del rel}; but this ambiguity can be reduced to a two-way ambiguity by the use of SSCs in combination with the known features on the left-hand side of the rule, since two combinations ([+ant +del rel] and [-ant -del rel]) can be ruled out. In general, the more features there on the right hand side of the rule (and hence the more ambiguous the underlying feature values, apart from pruning), the more likely it is that some combinations of those features can be ruled out. It is clear, then, that using SSCs considerably improves the Brute Force method.

Thus far, I have considered only the case where a single rule is un-applied, without regard for other rules, nor for the possible reapplication of the rule in question. The interaction of several rules results in a combinatorial multiplication: the number of feature values which must be instantiated in the course of analysis is (roughly) the product of the number of feature values which must be instantiated during the un-application of each rule. This combinatorial explosion is one of the major reasons it has seemed that the automatic un-application of phonological rules is a computationally difficult problem. This problem of multiple rule application will be the topic of the next section.

The effects of a rule which can re-apply to its own output can be even worse. Consider the following plausible consonant cluster simplification rule:

$$\text{C} \longrightarrow \emptyset \; / \; \text{C} \_\_\_\_ \text{C}$$

If this rule is un-applied to a surface form with a two-consonant cluster, the result will be an intermediate form having a three-consonant cluster. But if the rule is allowed to un-apply to this intermediate form, it can un-apply in two places to yield a five-consonant cluster, and so on ad infinitum! There are two ways of avoiding this problem: placing ad hoc limits on the application of deletion rules (which are the only rules that can cause such infinite application), or requiring that the forms derived by reverse application of phonological rules meet certain conditions, such as Morpheme Structure Conditions.

Morpheme Structure Conditions (MSCs) would be the most principled solution. Nonetheless, a

---

[3] An assumption here is that the features on the left- and right-hand sides of the rule are disjoint. Anyone who has taught phonology has seen students write rules like the following:

$$\begin{bmatrix} \text{C} \\ +\text{vd} \end{bmatrix} \longrightarrow [\text{-vd}]$$

/ (some environment)

The *+vd* specification on the left-hand side is redundant; without it, the rule applies vacuously to underlyingly nonvoiced consonants. The phonological literature as well contains many such rules with redundant specifications, but they can usually be reanalyzed to eliminate the redundancy. Of the few rules which resist reanalysis, most employ such debatable techniques as alpha switching variables or angled brackets. I leave it to phonologists to determine whether rules which necessarily employ the same features on both sides of the arrow actually occur in natural languages.

morphing program must rely on the linguist to write rules and conditions which in their combination will not cause problems. Nor are such interactions always obvious. For instance, the above rule could be written to delete consonants only at morpheme boundaries:

C ⟶ Ø / C ___ +C

Then if morphemes of a single consonant are allowed, MSCs would not prevent the rule from looping infinitely, endlessly postulating deleted morphemes. (I assume here that morpheme boundaries, unlike other parts of the environment, must be postulated as needed during un-application of phonological rules, since they are unlikely to be marked in surface forms. Clearly such postulation will have to be restricted. See Barton, Berwick and Ristad 1987, sec. 5.7, concerning problems caused by unrestricted postulation of segments which are null at the surface.) Furthermore, it has often been proposed that MSCs do not apply to the output of phonological rules (Kenstowicz and Kisseberth 1977, chap. 3, and Anderson 1974, chap. 15). Hence, ad hoc limits on rule reapplication will be needed, even with MSCs.

One might hope that Word Structure Conditions (WSCs) would have the desired effect in constraining rules. However, since WSCs apply to surface forms, they cannot help. In fact, from one perspective a consonant deletion rule exists in order to bring a nonconforming underlying representation into conformance with a WSC; hence the un-application of such a rule necessarily results in an intermediate form violating the WSC.

WHEN MUST FEATURES BE INSTANTIATED?

In this section I explore an approach to the problem of multiple rule interaction and the resulting combinatorial explosion. I will argue that features altered by rules can usually be left un-instantiated (at least until lexical lookup), thereby avoiding the combinatorial effects otherwise inherent in multiple rule application. The question then is, Under what circumstances do features actually need to be instantiated during un-application of a rule?

As a first approximation, if one rule assigns a value to some feature, while the environment of an earlier rule refers to that same feature, it may be necessary to instantiate the feature values altered by the later rule. I will refer to the situation where such instantiation becomes necessary as "interference" between the two rules.

We can be more precise about when interference occurs, since two rules do not interfere if the second rule can only alter the feature in an environment in which the first rule could not apply.[4]

Before giving a more explicit definition of when two rules interfere, I present a definition of *phonological unification:*

> Let $X = X_1 \ldots X_a \ldots X_i$ and $Y = Y_1 \ldots Y_b \ldots Y_j$ be two non-empty phonological sequences (i.e. sequences of segments, each segment being a set of distinctive features). Then X and Y *p[honologically]-unify,* with X_a and Y_b *corresponding,* iff there exists a phonological sequence $Z = Z_1 \ldots Z_g \ldots Z_k$ such that Z_g is the unification of X_a and Y_b, and all the other segments of Z are the unifications of the respective segments of X and Y (where X and Y may be extended to the left and/or right as necessary by the addition of empty segments).

Consider then the following two rules:[5]

A ⟶ B / C _ D

E ⟶ F / G _ H

---

[4]A concrete example of a situation where there is no interference, despite the fact that the second rule alters a feature referred to by the first rule, is the following two hypothetical rules:

$$\begin{bmatrix} C \\ -vd \end{bmatrix} \longrightarrow [+asp] \,/\, \_\_\_\ V$$

C ⟶ [-vd] / ___ #

Since the second rule devoices consonants only word finally, it will never interfere with the first rule, which refers to voiceless consonants only in pre-vocalic position. I asume here that there is no rule of word-final vowel deletion ordered between these rules; see fn. 6.

[5]I assume the features on the two sides of each rule are disjoint; see fn. 3.

In order to un-apply the first rule to a form, the values of the features given in A, B, C, and D must be known in that form, so that they can be matched against the values required by the rule. Interference occurs when the second rule alters any of the features of A, B, C, or D in an environment compatible with the application of the first rule. More specifically, let W (the output of the first rule) = C (A ∪ B) D, where (A ∪ B) = the unification of A and B; and let w_i be a segment of W such that w_i includes one or more of the features of F (i.e. the features altered by the second rule, not necessarily with the values specified in F). Then the second rule will interfere with the first if G E H p-unifies with W such that the segment E corresponds to w_i.

If we restrict our attention to the case where the output of the second rule contains but a single feature, the interference just described corresponds to one of two types of rule interactions in phonological theory: counterbleeding and counterfeeding interactions. To see why, suppose the order of application of the two rules were reversed. Then the rule E ⟶ F (now the first rule to apply) assigns certain values to the feature F, while the other rule relies on a certain value of that feature being present in its environment. Furthermore, the two environments are compatible (p-unifiable), by hypothesis. Then if the first rule (E ⟶ F) assigns the required value to F, it feeds the second rule (A ⟶ B). Similarly, if the first rule assigns to F the opposite of the required value, it bleeds the second rule. Since the actual rule order is the reverse, the rules stand in either a counterfeeding or a counterbleeding relationship.

The reason for restricting attention to one feature of F at a time, is that the rules may stand in a counterfeeding relationship with respect to one feature, but a counterbleeding relationship with respect to another feature. In such a case, the pair of rules as a whole will be in a counterbleeding relationship. (There may also be features in F which do not cause interference.)

It is significant that counterfeeding and counterbleeding rule orders are precisely those rule orders which are opaque (cf. Kiparsky 1971). In other words, the features altered by the second rule will have to be instantiated just in case the two rules are opaquely ordered.[6] Crucially, opaque rule orderings appear to be quite rare in natural language (Kiparsky 1971). (More precisely, rules which are opaquely ordered tend to be lost, reordered, or reanalyzed, so that opaque orderings are unstable. As a result, they tend to be rare.)

The fact that interference only occurs with opaque rule orderings suggests a better method of rule un-application than the Brute Force approach: instantiate features in a rule only if they (potentially)

---

[6]This description of potentially interfering rules is complicated by the fact that a rule ordered between two other rules can change their interaction. Consider the following two rules:

$$k \longrightarrow k^h \,/\, \_\_\_ \begin{bmatrix} V \\ +high \\ +stress \end{bmatrix}$$

$$\begin{bmatrix} V \\ -stress \end{bmatrix} \longrightarrow [-high] \,/\, \_\_\_ \begin{bmatrix} V \\ -high \end{bmatrix}$$

Since these rules are not p-unifiable (the [-stress] feature requirement in the second rule not being unifiable with the [+stress] feature in the first rule), the rules cannot interfere as they stand. But now let a second rule be introduced, ordered between these two, so that the new set of rules is the following:

$$k \longrightarrow k^h \,/\, \_\_\_ \begin{bmatrix} V \\ +high \\ +stress \end{bmatrix}$$

$$V \longrightarrow [-stress] \,/\, \_\_\_ \; C\,V\,C \begin{bmatrix} V \\ +stress \end{bmatrix}$$

$$\begin{bmatrix} V \\ -stress \end{bmatrix} \longrightarrow [-high] \,/\, \_\_\_ \; C \begin{bmatrix} V \\ -high \end{bmatrix}$$

The first rule is now (potentially) interfered with by both of the other rules: the second rule alters the stress on the vowel, and the third rule alters the height of that vowel in an environment which may now be compatible with that of the first rule because of the destressing rule.

For an intermediate rule to alter the interaction of two other rules, the intermediate rule must be p-unifiable with both the other rules (otherwise it could not operate on any forms that both the other rules operated on); and it must change the value of the feature(s) on the first rule which block p-unifiability with the third rule into a value(s) compatible with the third rule, i.e. feed the third rule.

interfere with an earlier rule. Instead, when a rule is un-applied, simply mark the features it changes as uninstantiated (i.e. of unknown value). In practice, this will usually result in a large savings in search space due to the rarity of opaque rule orderings. It will still be necessary to instantiate these "empty" features prior to lexical lookup, but this is clearly a lesser problem, since the effects are not multiplicative (and the instantiation can again be restricted by reference to SSCs).

However, in the next section I will suggest an even better approach, which takes advantage of the fact that even though two rules interfere potentially, the interference may not arise in every word in which one or the other of the rules applies. (In fact, one can imagine that in a language having potentially counterbleeding or counterfeeding rules, it might be the case that no words actually meet the structural description of both rules.)

THE LAZY APPROACH: INSTANTIATING FEATURES ONLY WHEN NECESSARY

The strategy of the lazy approach should by now be clear: postpone instantiation of feature values altered by phonological rules until those values are actually needed, either by lexical lookup or in order to un-apply another rule (i.e. when all the instantiated features of a form match a rule, but the values of one or more uninstantiated features in the form are also specified by the rule). When features are instantiated, such instantiation may again be restricted by the SSCs. In effect, then, un-application of a phonological rule produces archiphonemes,[7] so that features are instantiated only when absolutely required. Assuming that opaque rule orders are as rare as phonologists have claimed, and that words in which both members of a pair of opaque rules apply are even rarer, this will greatly reduce the computational complexity of general computational morphology.

[7]I assume that all features start out instantiated in surface forms, even "irrelevant" features. If they were not, it would be impossible on examining a given un-instantiated feature to know whether it has become un-instantiated during the course of the derivation and therefore is a candidate for instantiation, or whether it is an irrelevant feature for a particular segment and therefore could not trigger the rule in question. Alternatively, one could keep track of which features have become un-instantiated during the (un-)derivation.

CONCLUSIONS

In summary, I have shown that one of the combinatorial difficulties which would appear to make implementation of general morphing programs impractical is the ambiguity of feature values arising during un-application of phonological rules. But in fact those ambiguous values are needed later in the derivation only in the case of opaque rule orderings. This apparent difficulty can therefore be dealt with by delaying the instantiation of features which have become un-instantiated until they are actually required. Since opaque rule orderings are relatively rare, this results in a considerable savings in search space against the alternative of immediately instantiating all features altered by rules. Delayed instantiation also represents a savings in search space against the alternative of instantiating only those features whose values may be required by another rule, since not all words will meet the structural description of both rules of an opaque pair.

REFERENCES

Anderson, Stephen R. 1974. *The Organization of Phonology.* New York: Academic Press.

Anderson, Stephen R. 1988. "Morphology as a Parsing Problem." In *Morphology as a Computational Problem,* ed. Karen Wallace, 1-21. UCLA [Linguistics Department] Occasional Papers no. 7, Working Papers in Morphology.

Barton, G. Edward, Robert C. Berwick, and Eric Sven Ristad. 1987. *Computational Complexity and Natural Language.* Cambridge, Mass.: MIT Press.

Brandon, Frank R. 1988. "FONOL: Phonological Programming Language." Unpublished computer file.

Kay, Martin. 1977. "Morphological and Syntactic Analysis." In *Linguistic Structures Processing,* ed. Karen Wallace, 131-234. Amsterdam: North Holland.

Kenstowicz, Michael, and Charles Kisseberth. 1977. *Topics in Phonological Theory.* New York: Academic Press.

Kiparsky, Paul. 1968. "Linguistic universals and linguistic change." In *Universals in Linguistic Theory,* ed. Emmon Bach and Robert Harms, 171-202. New York: Holt, Rinehart and Winston.

Kiparsky, Paul. 1971. "Historical Linguistics." In *A Survey of Linguistic Science,* ed. William Dingwall, 576-642. College Park: University of Maryland.

Kisseberth, Charles. 1970. "On the Functional Unity of Phonological Rules." *Linguistic Inquiry* 1:291-306.

Koskenniemi, Kimmo. 1984. "A general computational model for word-form recognition and production." *COLING-84* 178-181.

Stanley, Richard. 1967. "Redundancy rules in phonology." *Language* 43:393-436.

Various, 1983. *Texas Linguistic Forum* no. 22. Department of Linguistics, University of Texas at Austin.

Weber, David J., H. Andrew Black, and Stephen R. McConnel. 1988. *AMPLE: A Tool for Exploring Morphology.* Occasional Publications in Academic Computing. Dallas: Summer Institute of Linguistics.