Two Theories of Morphology, One Implementation

Mike Maxwell

Contents:

Abstract
1. Introduction
2. Affixes-as-Morphemes vs. Affixes-as-Rules
3. One vs. Many Underlying Forms
   3.1 Inadequate Theories of Phonology
   3.2 Intermediate Stages of Analysis
   3.3 Suppletion
4. Conclusions

Appendix 1: Hermit Crab's Theory
   1. Introduction
   2. Basic Perspective
   3. Morphological Rules
   4. Phonological Rules

Appendix 2: AMPLE and Hermit Crab
   1. Introduction
   2. Root Dictionaries
   3. Affix Dictionaries
   4. Tests

Appendix 3: How Are Allomorphy Rules Applied?
   1. Introduction
   2. Ordering of Allomorphy Rules
   3. Interaction of Allomorphy Rules with Phonological Rules
   4. Interaction of Allomorphy Rules with Morphological Rules
   5. Conclusion

Endnotes
References

Editor's note: This paper was originally presented at SIL's General CARLA Conference, 14-15 November 1996, Waxhaw, NC. CARLA, for Computer-Assisted Related Language Adaptation, is the application of machine translation.
techniques between languages that are so closely related to each other that a literal translation can produce a useful first draft. The morphological parsing program AMPLE is often used in CARLA applications.

Abstract

Theories of morphology have been classified as Item-and-Arrangement (in which both roots and affixes are treated as morphemes), or Item-and-Process (in which roots are morphemes, but affixes are rules). I will show that in reality, a description using affixes-as-morphemes (Item-and-Arrangement morphology) can be mapped into a single representation.

A different classification of morphological theories is based on whether all allomorphs are listed in the lexicon, or whether phonologically conditioned allomorphs are derived from a single listed form. I show that in reality, derivational theories incorporate a device (allomorphy rules) which can do virtually the same work as listing the phonologically conditioned allomorphs. In fact, it is possible to mechanically map a description with multiple listed allomorphs into a description with single underlying forms and allomorphy rules.

The implication of these two points is that a single computational implementation can serve a variety of theoretical approaches.

In appendix 1 and appendix 2, I explore as a case study the degree to which the particular Item-and-Arrangement notation of AMPLE corresponds to the Hermit Crab implementation of Item-and-Process morphology. Appendix 3 discusses some subtleties in the use of allomorphy rules as a replacement for multiple underlying forms.

1. Introduction

Theories of morphology are commonly classified as being either Item-and-Arrangement (in which both roots and affixes are treated as morphemes), or Item-and-Process (in which roots are morphemes, but affixes are rules). The distinction is undoubtedly important from a theoretical point of view, e.g. as an explanation for language universals. But from another point of view, the difference is less important than it might appear; I claim there is a straightforward mapping from an Item-and-Arrangement description to an Item-and-Process description (although not necessarily in the reverse direction). This has an interesting implication from a practical viewpoint: an Item-and-Arrangement description can be mechanically mapped into an Item-and-Process description. Thus, if there is a computational implementation of Item-and-Process morphology, it can be used to process an Item-and-Arrangement description in an appropriate notation.
Linguists have sometimes characterized the contrast between Item-and-Arrangement and Item-and-Process morphology in another way, as a choice between listing all allomorphs on the one hand, and deriving phonologically conditioned allomorphs from a single underlying form on the other. In the second section of this paper, I show that in reality, current theories of Item-and-Process morphology retain an "escape mechanism" in the form of allomorphy rules to treat phonologically conditioned allomorphs which cannot be readily derived by ordinary phonological rules. Because allomorphy rules are able to model virtually any kind of phonologically conditioned allomorphy, it is possible to mechanically transform a description with multiple underlying forms into a description with single underlying forms plus allomorphy rules. Again, the implication of this mapping is that a computational implementation allowing the use of allomorphy rules can be used to process both single and multiple underlying form descriptions.

The organization of this paper is as follows. Section 2 discusses the Item-and-Arrangement vs. Item-and-Process distinction as one of affixes-as-lexical items vs. affixes-as-rules, supporting the claim that there is a mapping from the former to the latter. Section 3 discusses the alternative characterization of Item-and-Arrangement morphology as a multiple underlying form model, and Item-and-Process morphology as a single underlying form model. The "escape mechanism" of allomorphy rules is introduced, and it is argued that whatever the theoretical reasons may be for preferring single underlying form models, there are a number of reasons to allow languages to be described from the viewpoint of multiple underlying forms. The mapping from descriptions having multiple underlying forms into descriptions with single underlying forms plus allomorphy rules is then described.

Section 4 discusses the advantages to be gained by making use of these mappings in the context of computational implementations of morphological models.

The first two appendices constitute a case study in this mapping: appendix 1 documents the capabilities of a particular Item-and-Process morphology program, Hermit Crab, while appendix 2 describes some differences between this program's capabilities and those of an Item-and-Arrangement program, AMPLE.

Appendix 3 explores some further issues in the mapping from multiple allomorphs to allomorphy rules.

2. Affixes-as-Morphemes vs. Affixes-as-Rules

In a seminal paper, Hockett (1954) divided theories of morphology into two classes: Item-and-Arrangement and Item-and-Process.\[1\]

Under an Item-and-Arrangement theory (henceforth "IA"), roots and affixes are both treated as morphemes, with at least one allomorph of each stored in the lexicon. This is the position of such works as Lieber (1980), Di Sciuollo and Williams (1987), and Halle and Marantz (1993), to name a few. (The question of whether all allomorphs are stored in the lexicon, or whether phonologically conditioned allomorphs are derived by phonological rules, is a separate issue, although often confused with this issue; see section 3.) In contrast, under an Item-and-Process
theory (henceforth "IP"), only roots are morphemes, and therefore only roots are listed in the lexicon. Affixes are processes—another term would be "morphological rules"—and exist in a separate component of the grammar. This is the position of Aronoff (1976), Zwicky (1985) and Anderson (1992), among others.

With this view of the distinction between IA and IP models of morphology, consider how an IA analysis might be represented in an implementation of an IP model. I will claim that it is possible to directly map an IA description into an IP description.

For concreteness, consider the attachment of the English inflectional suffix -s to a plural noun. In IP morphology, this might be expressed by the following rule:

\[(1) [X]N \rightarrow [X s]N\]

That is, the rule of -s inflection simply attaches the phoneme s to the end of whatever phonological material the noun consists of, here represented by the variable X. In such simple cases of suffixation (or prefixation), there is an obvious mapping between an IP analysis and an IA analysis. To go from an IP description to an IA description, it is merely necessary to view the phonological material attached by the rule as if it were the phonological shape of a morpheme in the lexicon, and vice versa for the reverse mapping. The following lexical entry (expressed here in a generic notation) might be used to represent this affix. (The "SUBCATEGORIZATION: N" field represents the fact the input to the rule is a noun, while the "SYNTAX: N" field represents the fact that the output of the rule is a Noun. In this example, the latter field could have been left off, with the understanding that the output part of speech is identical to the input part of speech unless explicitly changed, but I have left it in for illustrative purposes.)

\[(2) PHONOLOGY: /s/\]

SYNTAX: N

SUBCATEGORIZATION: N__

The translation from the lexical entry in (2) to the rule in (1) is transparent. In fact, rules like that in (1) can be generated mechanically from lexical entries like that in (2).

Affixation can also take such forms as infixation, circumfixation, simulfixation, reduplication, and even subtraction. One of the traditional arguments in favor of IP morphology is that such nonconcatenative morphology cannot be represented in IA terms. Recent work in nonlinear phonology has weakened the force of many of these arguments; reduplication of the first syllable of the stem, for instance, can be modeled in IA morphology as the prefixation of an empty syllable, with subsequent filling of the syllable's structure by spreading from the stem (McCarthy and Prince 1990). Anderson (1992, chapter three) provides an overview of these issues, making the claim that at least some of the classical arguments against IA morphology are still valid. But from the standpoint of this paper, such arguments are irrelevant, as we are concerned not
with mapping from an IP morphological description to an IA description, but with the reverse mapping. Hence if certain types of affixation processes cannot be modeled in IA morphology, that is irrelevant, so long as any valid IA affix lexical entry can be mapped over to an IP rule. This is clearly the case for prefixes and suffixes. I am not aware of a generally agreed-on IA notation for other sorts of affixes. I will therefore of necessity leave the details of the mapping of nonconcatenative morphology from IA to IP open, although it seems likely that it would present no more difficulties than simple prefixes and suffixes.

In summary, I claim that an IA description can be mechanically mapped into an IP description. This has an important implication for computational implementations, e.g. morphological parsers: if a user is presented with a user interface which allows him to create an IA description, that description can be mapped internally into an IP representation. One advantage of separating the user interface from the internal "engine" in this way is re-use of the engine. Assuming the internal engine to be at least as complex as the user interface, this represents a considerable savings in program writing and maintenance. Moreover, should the IA model prove to be inadequate to the needs of a given language, the grammar that has been written up to that point can be retained; translating the grammar into the more adequate IP model is as simple as switching the interface. Finally, if such a change in models becomes necessary, the fact that the internal grammar description is (initially) unchanged may make it easier to retrain the user in the IP model, since he can view the same grammar in both IA and IP notations.

3. One vs. Many Underlying Forms

A second way in which the terms 'Item-and-Arrangement' and 'Item-and-Process' have been used to distinguish differing approaches to morphology involves the representation of allomorphy: are allomorphs listed or derived by phonological rules? (See e.g. Bybee 1985, and Pike and Pike 1982 for this usage of the terms.) Since this is not the sense intended by Hockett's original distinction, I will refer to models which assume all allomorphs to be listed in the lexicon (corresponding to the meaning of "Item-and-Arrangement" in Bybee 1985 and in Pike and Pike 1982) as "Multiple Underlying Form" models (henceforth, "MUF"), and to models which assume a single underlying form in the lexicon, with allomorphs being derived by phonological rules, as "Single Underlying Form" ("SUF") models.

An MUF model, then, assigns a fundamental status to allomorphs: all allomorphs (up to, but not including, allophonic variants) are stored in the lexicon, and a particular allomorph is chosen for lexical insertion based on the phonological and/or morphosyntactic environment. In the context of modern theories of linguistics, one may imagine the choice of allomorphs to be a constraint satisfaction problem, in which all the constraints of every allomorph making up the word must be satisfied. This view was for a time the dominant approach under American structuralism, and it has experienced a resurrection (in modified form) in work in Natural Generative Phonology (Hooper 1976), and later work on constraint-based phonology by Bird (1995) and others.

Under SUF theories, on the other hand, only one underlying form of a morpheme is stored in the lexicon; apart from suppletion, other allomorphs are derived by the application of phonological rules. In (American) structuralist linguistics, morphophonemic rules were of uncertain theoretical status (and allophonic rules were by definition not involved in allomorphy); hence MUF
morphology was dominant until the rise of generative linguistics. Early generative linguistics did not distinguish a separate morphological component, but the combination of lexicon, syntax, and phonology essentially operated under the assumptions of SUF morphology. That is, a single form of a morpheme was stored in the lexicon, and the phonology (which included a largely ignored component of "readjustment" rules, as well as ordinary generative phonological rules) derived the appropriate surface form. Beginning with Chomsky's "Remarks on nominalization" (1970), a distinct component for morphology was reintroduced, but the assumption of a single underlying form was retained, with allomorphs being derived by the phonology (see e.g. the discussion in Kenstowicz and Kisseberth 1979, chapter six). In summary, the SUF approach is now well accepted by generative linguists.

To anticipate the conclusion of this section, I will argue that whatever the advantages of SUF morphology may be in theory, in practice it is necessary to allow for the description of languages using an MUF approach. The first two subsections below present the reasons for this pragmatic conclusion. The third subsection suggests that SUF theories in fact provide an "escape" mechanism which allows the translation of an MUF description into an SUF description in a way which is both mechanical and transparent to the user. The implication is that while an underlying parsing engine may implement SUF morphology internally, it can in fact process descriptions couched in terms of either SUF or MUF morphology, with a minor amount of preprocessing.

### 3.1 Inadequate Theories of Phonology

The central claim of SUF morphology is that phonologically conditioned allomorphs result from the application of phonological rules to a single underlying form in diverse environments. However, it is sometimes the case that allomorphs which are clearly conditioned by the phonological environment cannot be straightforwardly generated by current theories of phonology.

Before turning to some examples, I should observe that this is a practical question, not a theoretical one. If we had the correct theory of phonology, the processes in question would presumably be readily statable (at least that is the working hypothesis of linguists who subscribe to SUF morphology). Unfortunately, a descriptive linguist may not be able to afford the luxury of waiting for the correct theory to arrive! Even when the correct theory has been developed and become generally accepted by theoretical linguists, it may not be feasible to retrain the descriptive linguist, or the potential readers of his description, in the latest theory. In short, it may be expedient to allow a description of the phenomenon in other than the most current theory.

Consider an example of phonologically conditioned allomorphy which resists easy description by phonological rules. The verbal person-marking prefixes of Tzeltal are given in the following table (the same prefixes are also used on nouns to mark the person of possessors):

<table>
<thead>
<tr>
<th>1st. person</th>
<th>_C</th>
<th>_V</th>
</tr>
</thead>
<tbody>
<tr>
<td>h-</td>
<td>k-</td>
<td></td>
</tr>
</tbody>
</table>
The conditioning is completely phonological, but it would be difficult to formulate an appropriate phonological rule to derive the allomorphs of the first and third person prefixes, given the currently accepted constraint that a phonological rule can affect only one node of an autosegmental feature structure. This is clearest in the case of the third person prefix. Suppose that the underlying form is s-. Then we require a rule to turn /s/ into /y/ before a vowel. Under the Halle-Sagey model of feature geometry (Halle 1992; see also Kenstowicz 1994, chapter nine), we might attempt to formulate the rule as one deleting the feature [-consonantal] from the root node, thus converting the sound from a fricative to a glide. But it is not clear how the feature [+strident] is to be eliminated (is this done automatically because there are no [+strident - consonantal] sounds?), or how the values of the features [voiced], [anterior] and [distributed] are changed. Conversely, we might assume y- to be the underlying form, with a rule turning /y/ into /s/ before a consonant. This rule is at least more plausible (there are no instances of /y/ before a consonant in Tzeltal, while there are numerous words containing /s/ followed by a vowel); but again, it is not clear how to effect the change. Why, for instance, does the /y/ turn into a /s/ instead of a /ʃ/, which would be closer in feature content?

No doubt there are answers to these questions, perhaps within some existing theory of phonology. My point is not that the rules cannot be formulated, given the correct theory. Rather, my point is that we do yet not know which (if any) theory is correct. Even if all phonologists agreed tomorrow on a theory which would explain the Tzeltal prefixes, there are numerous cases of phonologically conditioned allomorphy in other languages which would doubtless remain unexplained. As a practical matter, then, an implementation should allow the user to represent such situations using multiple allomorphs.

It must be noted, however, that this problem is less of a factor to the extent that the phonological description allows the statement of unnatural rules. That is, the less attention one pays to the restrictive universals postulated by phonologists, the easier it will be to state such allomorphy using phonological rules. (The allomorphy can easily be derived by classical generative phonological rules, of the type used in The Sound Pattern of English, for instance.) This is perhaps a case for not using an up-to-date theory!

### 3.2 Intermediate Stages of Analysis

But suppose for a moment that we had the correct theory of phonology, so that the objections of the previous section lost their force; that is, suppose that all phonologically conditioned allomorphs could be derived using phonological rules. I maintain that it would still be desirable for a practical implementation of a morphological parser to allow the listing of allomorphs, i.e. to allow MUF morphology.

<table>
<thead>
<tr>
<th>2nd. person</th>
<th>a-</th>
<th>aw-</th>
</tr>
</thead>
<tbody>
<tr>
<td>3rd. person</td>
<td>s-</td>
<td>y-</td>
</tr>
</tbody>
</table>

Table 1: Tzeltal Prefixes
Consider a field linguist who has begun work with a hypothetical unanalyzed language. For concreteness, let us assume the language has an agglutinating morphology and a phonology similar to that of Turkish.

In such a language, a number of phonological processes, none of them difficult to state in current phonological theories, affect affixes. For instance, suffix vowels might agree in the feature [back] with the vowel to their left, while high vowels might agree in the feature [round] as well. There might also be, as in Turkish, a process of word-final consonant devoicing, as well as other processes of more limited generality.

Thus, each affix is subject to a number of phonological processes, in some cases resulting in a large number of allomorphs.

Some affixes may be exceptions to some of these otherwise general processes. For example, in the Turkish suffix -Iyor 'momentary action', only the first vowel undergoes harmony. This suffix is also an exception to a rule of vowel deletion (see Zimmer 1970).

The field linguist doing the preliminary analysis of such a language might not notice at first the generality of the phonological processes, or he might notice the generality but be unable to account for the exceptions. He might need the help of a consultant, when one is available, to produce an account of allomorphy based on phonological rules. Nevertheless, before he arrived at a complete grasp of the (morpho-)phonology, the linguist might wish to process texts--parsing words into their component morphemes, for instance. The statement of allomorphy by means of multiple underlying forms would allow the linguist to do such text processing at this earlier stage of understanding, while still supporting the goal of eventually accounting for the allomorphy with phonological rules.

I therefore claim that as a practical matter, a system for morphological description should allow the statement of allomorphy using an MUF model, particularly during the earlier stages of analysis.

3.3 Suppletion

'Suppletion' refers to the existence of allomorphs which cannot be reasonably derived from underlying forms by phonological rules, either because the conditioning is not phonological, or because (at least two of) the forms to be related differ too greatly from each other to be derived from a common base form by plausible rules. An example is the English indefinite article, with allomorph an before vowels, and a elsewhere. There is no plausible sequence of phonological rules in English that would result in the epenthesis of n before vowels, or in the deletion of n before consonants; rather, the alternation between n and zero is confined to this particular morpheme. (The indefinite article happens to be a clitic (which poses interesting problems for the delineation between morphology and syntax), but a similar point can be made with affixes.)

Consider for instance the Tzeltal suffixes -hib 'place or instrument nominalizer' and -hom 'agentive, agentive nominalizer' (Slocum 1948). Both have an allomorph without the initial h, but the conditioning environment is different: for -hib, the allomorph -ib appears after a glottalized
consonant or l, with -hib elsewhere; for -hom, the allomorph -om appears after affricates or vowels, and the -hom allomorph elsewhere. Not only would it be difficult to derive both sets of allomorphs by general phonological rules, but the resulting rules would be very odd, since neither set of environments constitutes a natural class.

Another example from Tzeltal of suppletion is the verbal suffix -eh 'perfective aspect'; this has an allomorph -oh which appears after monosyllabic stems, while -eh appears elsewhere. There are several other Tzeltal suffixes having approximately the shape of one or the other of these allomorphs, but none which undergoes this particular alternation. It therefore seems unlikely that this alternation should, or even could, be accounted for by a general phonological rule, despite its phonological conditioning.

A number of other phonologically conditioned allomorphs in various languages for which straightforward phonological rules seem improbable are given by Carstairs (1987, table 1.1 page 21) and Spencer (1991: 121).

The existence of phonologically conditioned allomorphy which will not readily succumb to treatment by general phonological rules has not escaped the notice of SUF proponents. Chomsky and Halle (1968) proposed a solution in which a distinct class of rules, "readjustment rules," apply before phonological rules. Chomsky and Halle's theory included several types of readjustment rules; we will be concerned here with those that are essentially restricted to applying to specific morphemes, and have therefore been referred to as "allomorphy rules." The following are some examples:

\[
(3) \quad C \rightarrow C^* \left/ \begin{array}{c}
\varepsilon \\
\text{su}
\end{array} \right. = C^* \\
\text{where } C \text{ and } C^* \text{ are both coronal or both noncoronal}
\]

\[
(4) \quad \text{Rime } \rightarrow \left/ \begin{array}{c}
\text{u} \\
x \text{ past}
\end{array} \right. = \left/ \begin{array}{c}
\text{shall, will, can, stand}
\end{array} \right.
\]

\[
(5) \quad t \rightarrow \left/ \begin{array}{c}
\text{voice}
\end{array} \right. = \left/ \begin{array}{c}
\text{mit}+\text{ive} \\
\text{vert}+\text{ive}
\end{array} \right. (\text{where } 'V' \text{ stands for an indeterminate vowel})
\]

\[
(6) \quad -fy \rightarrow -fic / \_\_ -Ation
\]

((3) and (5) are Chomsky and Halle's (1968) examples (3) and (2) respectively, page 238; example (4) is Halle and Marantz's (1993) example (10a), page 128; (6) is based on the discussion in Aronoff (1976) chapter five, and is given in orthographic, not phonological, form. See also Gussmann (1980, section 2.5) for further examples of allomorphy rules.)

It may not be apparent at first glance that these rules apply only to specific morphemes. Rule (3), for instance, appears to refer to a more or less phonetically statable environment. But in fact that environment is so specific that it affects the allomorphs of only two morphemes: the prefixes ad- / ab-, and sus-/ sub-. Likewise rule (4) applies to only the four stems listed in the rule. Rule (5) explicitly targets the morphemes =mit and =vert, making use of boundary markers to limit its
application to these two morphemes, and rule (6) is even more specific, applying as it does to one particular morpheme in the environment of another particular morpheme. These examples thus demonstrate an important point: allomorphy rules in reality represent the retention in SUF morphology of a form of MUF morphology, disguised as rules.

Let me make this more explicit: I claim that a lexical entry with multiple allomorphs can be mechanically converted into a lexical entry with a single underlying form (generally, one of the allomorphs), together with one or more allomorphy rules to convert that underlying form into the other allomorphs. In some cases, the use of allomorph rules may require that diacritic (rule) features be assigned to the morphemes which are to undergo a certain process. For instance, Halle and Marantz's rule given above as (4) is inadequate as it stands, in that it will incorrectly apply to the verbs *can* 'to preserve by canning' and *will* 'to deed over effective on one's death.' In these examples, it is the morphemes that undergo the allomorphy rule which require the diacritic feature; there may also be cases where the conditioning morpheme must be marked by a diacritic. At any rate, the use of diacritic features makes it possible to directly encode in an SUF morphology anything which could be encoded in an MUF morphology.

In summary, SUF theories retain an "escape mechanism" in the form of allomorphy rules for cases of phonologically conditioned allomorphy which resist treatment by general phonological rules. Because allomorphy rules can refer to specific morphemes, it is possible to mechanically translate any MUF description into an SUF description by choosing one allomorph to be the underlying form, marking it with a unique diacritic feature, and deriving the remaining allomorphs in the appropriate environment using allomorphy rules sensitive to that diacritic feature. As I argued earlier, when it comes to computational implementations, one advantage of being able to represent one theory in terms of another is re-use of the parsing engine, at the cost of building a user interface to translate between theories. Allowing the user to resort to an MUF description also has the advantages outlined in the previous two sections, namely of making it possible to describe allomorphy which cannot easily be treated by current theories of phonology, and of allowing the field linguist to handle allomorphy in the earlier stages of his investigation, before the necessary phonological rules have become apparent.

4. Conclusions

The terms "Item-and-Arrangement" (IA) morphology and "Item-and-Process" (IP) morphology cover two distinctions which are sometimes confused. In the first usage, the question is whether affixes are to be treated as lexical items, on a par with roots and other "listemes," or whether affixes should be treated as rules (processes). I have shown that from a sufficiently abstract point of view, this is a non-issue: an IA description can be mechanically translated into, and stored as, an IP description. The implication for computational implementation is that a morphological parsing engine which implements Item-and-Process morphology can in fact be used for Item-and-Arrangement morphology by means of an appropriate user interface.

The second distinction sometimes intended by the terms "Item-and-Arrangement" vs. "Item-and-Process" is that between Multiple Underlying Form (MUF) models and Single Underlying Form (SUF) models. Again, from a sufficiently abstract point of view, an MUF description can be mechanically translated into an SUF model, and stored as such.
Much is gained by using a single parsing engine to implement varying theoretical models. One immediate advantage is the lesser amount of programming effort required, assuming the internal engine to be at least as complex as the user interface. Equally important, I would argue, are the advantages to the end user. It becomes possible for the user to gradually shift from one theoretical perspective to another in the course of a language program. Allomorphy can be modeled initially in terms of multiple allomorphs, each with a separate conditioning environment; as the phonological processes causing the allomorphy are clarified, separate allomorphy statements can be collapsed into one or more phonological rules. There is no need at any point to make a sudden and complete shift from MUF morphology to SUF morphology, nor if it is decided to switch from one framework to the other is it necessary to rewrite the entire grammar from scratch. Finally, as suggested in section 2, there is also a pedagogical advantage to being able to display a description in two theoretical frameworks, particularly to the user who is unfamiliar with one of those frameworks.

Appendix 1: Hermit Crab's Theory

1. Introduction

Hermit Crab is a computer program which can be used for both morphological parsing and generation. This appendix presents a brief overview of Hermit Crab and its theoretical perspective. The presentation is done in an informal way by presenting example rules in a notation which should be more or less familiar to linguists. While Hermit Crab's internal notation is somewhat different, it is not intended to be used by humans; the reader will have to take my word that the notations used in this appendix can be directly translated into Hermit Crab's notation. Further details are given in Maxwell 1991, Maxwell 1994, and Maxwell ms.

2. Basic Perspective

The general theoretical perspective of Hermit Crab is that of classical generative phonology, i.e. generative phonology as practiced between the time of SPE (Chomsky and Halle 1968) and the rise of autosegmental phonology. Phonological representations are treated as sequences of feature bundles, and phonological rules apply to such feature bundles by inserting feature values or copying feature values (rather than by spreading features). The feature system is user-defined. Rule strata can also be defined (as in Lexical Phonology). Rules of a given stratum apply in linear order (simultaneous rule ordering could be implemented as well). Individual phonological rules apply in left-to-right or right-to-left iterative fashion (simultaneous application could also be added). The morphological rules of a stratum apply first, then the phonological rules of that stratum; cyclic application could also be implemented.

The following two sections discuss morphological and phonological rules.

3. Morphological Rules
Hermit Crab takes an Item-and-Process view of morphology. That is, affixes are treated internally as rules, rather than as lexical items. A very simple rule attaching a verbalizing suffix might be expressed as follows:

$$[X_1]_N \rightarrow [1 \, ut]_V$$

In other words, the rule takes a noun stem (whose phonological content is represented by $X_1$) and attaches a suffix (represented by $ut$) to it; the resulting word has part of speech of verb. The use of the subscript 1 on the left-hand side of the rule and the numeral 1 on the right indicate that the noun's stem is simply copied across without change.

As stated above, Hermit Crab treats segments (phones, phonemes etc.) as feature bundles; hence the representation above is a simplification. In particular, the variable $X$ representing the stem's phonological content is actually a regular expression consisting of an empty feature bundle repeated from one to infinitely many times. Since an empty feature bundle matches against any feature bundle, the effect is that the $X$ matches against a stem of any length. The suffix itself, represented by the string $ut$ in the above example, is translated internally from such a string representation into a sequence of feature bundles. It is also possible to represent all or part of the suffix as a feature bundle externally. This might be desirable if many of the phonetic features of the suffix were determined by later phonological rules. For instance, if the backness and rounding features of the vowel were determined by vowel harmony, one might modify the above rule as follows:

$$[X]_N \rightarrow [X \, \text{high\_back\_vowel} \, t]_V$$

-- where high_back_vowel had been previously defined as a "natural class" bearing the appropriate phonetic features, e.g.:

$\text{high\_back\_vowel} \equiv [-\text{consonantal} +\text{syllabic} +\text{high} +\text{back}]$

(The empty feature bundle represented in the above examples by $X$ is actually defined as a natural class in the same way.)

In addition to segments whose phonological material is represented by strings or natural classes, morphological rules may insert boundary markers and diacritic (rule) features. This allows later rules (such as allomorphy rules) to distinguish individual morphemes.

Now consider infixes. For concreteness, suppose the infix -n- attaches after the first consonant + vowel of the stem. Such an affix might be represented as follows:

$$[C_1 \, V_2 \, X_3]_N \rightarrow [1 \, 2 \, n \, 3]_V$$

The natural classes $V$ and $C$ might be predefined as follows:

$V \equiv [-\text{consonantal} +\text{syllabic}]$
C = [+consonantal -syllabic]

As stated, this infixation rule would not apply to a stem which did not begin with a consonant plus a vowel. In some languages, if the stem is not parsable in such a way, the affix would instead apply as an prefix. Hermit Crab therefore allows the use of subrules of a given morphological rule, which apply following the familiar[12] "Elsewhere" principle of application (Kiparsky 1973).

Such a rule might be written as follows:

$$\begin{align*}
\left\{ C_1 \; V_2 \; X_3 \right\}_N & \rightarrow \left[ 1 \; 2 \; 3 \; 3 \right]_Y \\
\left\{ X_1 \right\}_N & \rightarrow \left[ n \; 1 \right]_Y
\end{align*}$$

The braces indicate that the two subrules belong to a single affixation rule, and apply disjunctively in the stated order (that is, the second subrule applies to a given word only if the first subrule is unable to apply).

Rules of reduplication may also be defined. The following rule, for example, indicates that the first CV(V) of the stem is reduplicated. The parentheses indicate that the second vowel is optional (that is, if there is a second vowel immediately following the first, it is reduplicated along with the preceding consonant plus vowel; otherwise the $V_3$ in the input and the corresponding $3$s in the output are ignored):

$$[C_1 \; V_2 \; (V_3) \; X_4]_Y \rightarrow [1 \; 2 \; 3 \; 1 \; 2 \; 3 \; 4]_Y$$

It is also possible to define morphological rules of simulfixation and truncation, as well as combinations of any of the simple rule types. A circumfix, for instance, may be defined by a rule which simultaneously attaches a prefix and a suffix. Finally, so-called zero morphemes are captured simply by a rule which copies its input to its output. Typically, a zero morpheme would be only one subrule of a given rule. Alternatively (as suggested by Anderson 1992), zero morphemes will often not correspond to any affixation process whatsoever, and therefore need not be parsed.

The examples thus far have dwelt on the phonological effects of affixation, plus the change in the part of speech. Morphological rules must also refer to such morphosyntactic features as tense, person, number etc. In general, there are two parts to this: an affixal rule may place requirements on the values of the morphosyntactic features belonging to the stem which it modifies, and an affix may supply morphosyntactic feature values of its own which override any feature values of the stem.[13] These two parts of the feature percolation convention are supplied in Hermit Crab morphological rules by two fields, the required features and (head) features fields. The required features must unify with the stem's features, while the head features are added to the stem's features to become those of the resulting word (with affixal features overriding any conflicting feature values of the stem). For instance, past tense suffixes in Cubeo (Tucanoan, Colombia) attach only to "dynamic" stems. This could be expressed by including the feature [+dynamic] in
the *required features* field of the past tense suffix rules, and the feature [+past_tense] (assumed for convenience to be binary) in the *head features* field of the rules.

It is sometimes desirable to require that a stem to which an affix rule is to apply not bear any value for a given feature. Hermit Crab uses a designated null value for this purpose.

In addition, *rule features* may be specified for such purposes as conjugation or declension classes; their behavior is similar to that of morphosyntactic features.

### 4. Phonological Rules

Hermit Crab also allows the use of phonological rules, making it possible to take an SUF view of morphology. Allomorphy rules are implemented as a subtype of phonological rules (although the discussion of appendix 3 suggests some changes may be necessary).

A simple phonological rule consists of an input, output, and left and right environments. The input and the output usually consist of a sequence of segments or natural classes, both of which are treated internally as feature bundles. Alternatively, either the input or output may be empty, resulting in a rule of epenthesis or deletion, respectively. The left and right environments are a sort of regular expression whose leaf elements may be natural classes, segments, or boundary markers, and whose nonterminal elements encode optionality and repetition. Finally, it is possible to require agreement between feature values in the input, output, and environment (i.e. so-called "alpha" variables are supported).

The following is an example of a rule of vowel harmony in which a vowel agrees with the roundness of the nearest vowel to its left:

\[
V \rightarrow [\alpha \text{ round}] / \left[ \begin{array}{c} V \\ \alpha \text{ round} \end{array} \right] C^* 
\]

The *natural classes* V and C are as defined above.

Phonological rules can also be sensitive to part of speech of the word to which they are applying (necessary e.g. to capture the facts of English stress assignment) and to rule features. This latter capability, together with the ability to include boundary markers in the left or right environment, allows allomorphy rules to be treated as a subtype of phonological rule. That is, it is possible to define rules which apply to specific morphemes or rules which apply in the environment of a specific morpheme, even if there are homophones to which the rule should not apply.

### Appendix 2: AMPLE and Hermit Crab

#### 1. Introduction

In this appendix, I investigate the degree of correspondence between the particular IA notation of AMPLE (as described in Weber, Black and McConnel 1988 (henceforth "WBM"), and in
Buseman et al. (1992), and the IP notation of the Hermit Crab morphological parser described in the preceding appendix. I am not concerned here with the detailed syntax of the two notations, but rather with the meaning of the two representations. The question is, which generalizations expressed in AMPLE's notation cannot be expressed in Hermit Crab's notation; and of these, which are likely to be important for linguistic purposes?

One major difference between the two programs is that AMPLE takes a "string" view of the world, while Hermit Crab takes a phonetic feature-based view. Thus, while it is possible to define natural classes of sounds in AMPLE, the definition is in terms of a set of strings (as suggested by the name used in AMPLE for such natural classes, namely "string class"). The class of alveolar sounds might, for instance, be defined in AMPLE as the set of strings \{t d n s\}. In contrast, Hermit Crab would define such a class in terms of the values of phonetic features, for instance [+coronal] (see appendix 1 for some examples); the set of segments (or phonemes) included in a class is a derivative notion. This is not to say that the translation from string classes to feature-based classes could not be automated, once the user has decided on a feature system and defined the features of segments.

In addition to classes of strings, AMPLE allows the use of particular strings in the environments of allomorph statements; these have a direct translation (assuming the strings consist of complete phonemes, which seems a reasonable assumption), since Hermit Crab allows the use of segments, not just feature bundles, in the environment of allomorphy rules (which are treated in Hermit Crab as a kind of phonological rule).

In addition to the way in which natural classes are represented in the two programs, there are a number of additional differences, of which the major divergence lies in the treatment of morphosyntactic features. The following sections examine these differences in more detail.

2. Root Dictionaries

Both AMPLE and Hermit Crab have a notion of a dictionary as being a list of lexical entries of roots (actually, stems, as there is no reason derived or even inflected stems could not be listed in such a dictionary). Within that general notion, each program has expectations about the information contained in those lexical entries. I will base my discussion of the contents of the fields of lexical entries on that in WBM chapter 11, section 3. The fields listed there, and their translation into Hermit Crab's conceptual view, are as follows:

2.1 Root

This field is used by AMPLE only to define the beginning of a lexical entry record, and is therefore irrelevant to this comparison. (Lexical entry records in Hermit Crab are delineated by brackets in a manner which need not concern us here.)

2.2 Allomorph

The treatment of allomorphy in AMPLE is considerably different from that in Hermit Crab. To summarize the discussion in the body of this paper, AMPLE represents an MUF approach to
allomorphy, whereas Hermit Crab takes an SUF approach. Nevertheless, there is good reason to allow the user to set up allomorphy statements apart from phonological rules (as argued above in section 3 of the body of this paper), and these allomorphy statements can be used to simulate an MUF model. One way to do this is to translate the allomorphs of an AMPLE lexical entry together with the conditions on their occurrence into allomorphy rules.

These conditions on allomorphs in an AMPLE lexical entry include 'morpheme environment constraints' and 'string environment constraints.' String environment constraints encode phonetic properties of the environment in which the allomorph is allowed to appear, or in which it must not appear. Positive string environment constraints have a direct translation into the environment conditions of Hermit Crab allomorphy rules. Negative string environment constraints, however, do not have a straightforward translation, and must be converted into a disjunction of environments.\[15\]

The translation of AMPLE's Morpheme Environment Constraints (MECs) into a Hermit Crab structure is a more complex issue. A MEC in which the environment may be separated from the allomorph being tested is equivalent to an AMPLE test using the FOR_SOME_LEFT or FOR_SOME_RIGHT operator, applied to the allomorph(s) in question. For example, the MEC

\[+/... \{\text{FUT}\}\]

is equivalent to the AMPLE test

\text{FOR\_SOME\_RIGHT} \text{RIGHT property is FUT}

(i.e. \((\exists x \text{ FUT}(x))\), ignoring the directionality); and the MEC

\[+/\sim ... \{\text{FUT}\}\]

is equivalent to the AMPLE test

\text{NOT FOR\_SOME\_RIGHT} \text{RIGHT property is FUT}

(i.e. \((\neg (\exists x \text{ FUT}(x))\), or equivalently, \((\forall x \neg \text{ FUT}(x))\)). Since the general tests subsume the specific MEC tests, I will defer the question of the equivalence between the AMPLE and Hermit Crab tests to the discussion in section 4 below.

An MEC in which the environment is obligatorily adjacent to the allomorph being tested is not directly translatable into Hermit Crab's notation as the latter currently stands. It may be that this capability should be added. For instance, the allomorphy rule given in example (6) and repeated here seems to require this power:

(7) \(-fy \rightarrow -fic / -Ation\)

This allomorphy rule clearly refers to specific morphemes, not just to any sequence of phonemes with the appropriate shape; and the morphemes in question must be adjacent, although it is
doubtful in this particular example whether -fy and -Ation could both occur in an English word without being adjacent. Thus, while the judicious use of boundary markers would probably succeed in preventing the unwanted application of this rule of English, one can imagine cases where that would not be sufficient (homophonous affixes with different allomorphs or conditioning properties, for instance). At this point, Hermit Crab's allomorphy rules do not have the capability of referring to the features (or glosses) of adjacent morphemes, but this could easily be added.

In addition to conditions on the appearance of an allomorph, AMPLE can assign Allomorph Properties to individual allomorphs (as opposed to the morpheme properties, which are assigned to all allomorphs of a given morpheme). There is no provision in Hermit Crab for assigning non-phonetic features in the output of allomorphy rules (nor to phonological rules in general); I will now attempt to show that this lack is a virtue.\(^{[16]}\)

There are several reasons why one might assign non-phonetic features to particular allomorphs:

1. to indicate a class of allomorphs of more than one morpheme, such that the appearance of all the allomorphs in the class is conditioned by some property in common;
2. to indicate that selected allomorphs condition a process which affects other morphemes; and
3. to allow dummy (zero) morphemes to condition allomorphs.

An example of (1) is given in WBM, section 13.4.6 (page 176), the essence of which is as follows. There is a set of morphemes which undergoes vowel shortening, and another set of morphemes which triggers this shortening process. The latter class is provided with a Morpheme Property 'foreshortens.' One way to ensure that morphemes of the first class appear with shortened vowels appear always and only before morphemes of the second class, would be to attach MECs to the allomorphs of the first class, for instance:

\[
\text{ma: } +/ - \{\text{FORESHORTENS}\}
\]

\[
\text{ma } +/ _{\{\text{FORESHORTENS}\}}
\]

The analysis given in WBM instead, uses a second morpheme property 'underlyinglong', and an allomorph property on all the shortened allomorphs 'foreshortened.' In addition, a test is added to ensure that if a morpheme has the property 'underlyinglong,' then if it appears before a morpheme with the morpheme property 'underlyinglong,' its allomorph with the 'foreshortened' allomorph property appears, otherwise an allomorph without this allomorph property appears.

I would argue that the WBM analysis is correct insofar as it allows a generalization to be made, namely that all morphemes with a particular property undergo an alternation.\(^{[17]}\) However, the generalization is captured in a linguistically inappropriate way, namely as an if-then condition on allomorph properties, rather than as a rule describing phonological structure. In Hermit Crab, the generalization would be captured instead by a phonological rule which shortens the vowels in morphemes appearing in the appropriate environment.\(^{[18]}\) That is, the alternation is treated in Hermit Crab as a phonological process affecting vowels, rather than as an alternation between
named allomorphs. The alternation is thus automatically restricted to exactly that portion of the morpheme which is relevant (preventing, for instance, the situation in which some morphemes have a shortened allomorph in the relevant environment, while others have a lengthened allomorph, or perhaps a devoiced allomorph, or even no allomorph at all in the shortening environment). Putting this differently, Hermit Crab allows the linguist to make generalizations directly using phonological structure, rather than indirectly using names of allomorphs.[19]

The situation in (2), where an allomorph property indicates that only certain allomorphs condition a process affecting other morphemes, might arise when a phonological process affects morphemes on both sides of a morpheme boundary. For instance, suppose there is a process in some language where a consonant is deleted in a certain environment, and that the preceding vowel lengthens (compensatory lengthening). Frequently this process will take place across a morpheme boundary, with the lengthened vowel preceding the boundary and the consonant being deleted after the boundary. This situation might be described in AMPLE by assigning an allomorph property 'deleted C' to allomorphs from which a consonant has been deleted, and conditioning the appearance of allomorphs with lengthened vowel by this allomorph property. While the AMPLE solution works, I again claim that it is the wrong way of looking at the problem, because it breaks the link between the phonological structure causing the vowel lengthening and the lengthening process itself. The allomorph property 'deleted C' is simply a name; there is nothing in that name to ensure that it appears only on allomorphs from which an initial consonant has been deleted, nor is there any way to ensure that it does appear on all allomorphs from which an initial consonant has been deleted. Putting this differently, the allomorph property is an arbitrary diacritic which has no inherent link to the phonological process its name describes. A better solution is to describe a process which at the same time deletes a consonant and lengthens a vowel.

I therefore conclude that, for these two cases, the uses to which allomorph properties have been put are better served by directly stating the phonological alternations involved.

The third case in which one might use allomorph properties (situation (3) above), is one in which the allomorph properties of a non-overt (zero, or dummy) morpheme condition allomorphs of an overt morpheme. Bloch (1947) makes a similar proposal in his analysis of English irregular (strong) verbs, if we interpret his inflectional classes as allomorph properties. For instance, Bloch analyzes the past tense of the verb take as take + a zero allomorph of the past tense, where the /e/ (orthographic a) becomes /u/ (orthographic oo) when the stem take appears in the environment before this zero allomorph. Translating Bloch's analysis into an AMPLE analysis, we would say that there is a zero allomorph of the past tense which bears an allomorph property, say causes ablaut, while the morpheme take has a morpheme property ablautable, and an allomorph took with allomorph property ablauted. We may then write a test which ensures that if a morpheme has the morpheme property ablautable, then the allomorph of that morpheme with the allomorph property ablauted always and only co-occurs with the (zero) allomorph of the past tense morpheme having the allomorph property causes ablaut.

While this is a possible analysis of strong verbs, it is perhaps not the most perspicuous analysis, in part because of its reliance on zero morphemes. In Hermit Crab, two other analyses are available. Without going into details, the first is an IP analysis in which a subrule of the
morphological rule for past tense ablauts the stem vowels and assigns a morphosyntactic feature such as [past tense]. The ablaut subrule would be applicable only to verbs bearing a special diacritic (rule) feature, while other subrules might apply to verbs bearing other rule features, and the 'elsewhere' subrule would apply to regular verbs (verbs unmarked by any of the relevant diacritic features). The other analysis possible under Hermit Crab is a lexical analysis, in which strong verbs would have explicit lexical entries for their past tense forms; by the principle of blocking (Aronoff 1976), these lexical entries would be chosen as past tense forms, blocking the application of an affix rule to the bare stems.

### 2.3 Categories

The view of categories (parts of speech) taken by AMPLE and Hermit Crab represents a divergence between the two programs. In AMPLE, a single lexical entry may bear multiple categories, whereas in Hermit Crab a lexical entry has a single part of speech. This difference may be readily bridged, however, by creating one copy of the AMPLE lexical entry for each corresponding part of speech, an expedient which may be necessary in AMPLE in any case if syntactic features are used (Buseman et al. 1992: 17). (See also the discussion of category pairs of affixes, below.)

Note that Hermit Crab does allow for a single lexical entry to have multiple subcategorizations (i.e. lists of syntactic complements).

### 2.4 Etymology

The etymology field corresponds to Hermit Crab's gloss field. (Note that the Hermit Crab gloss field can be used for an etymological form just as easily as it can for a gloss; compare the discussion of the dual usage of this AMPLE field in WBM: 128.)

### 2.5 Morpheme Properties

Like Allomorph Properties, Morpheme Properties are designed to express cooccurrence restrictions (WBM: 128); but unlike Allomorph Properties, Morpheme Properties are assigned to all allomorphs of a given morpheme. Their use encompasses both morphosyntactic features and diacritic (rule) features.

In a later version of AMPLE, "features" were added (Buseman et al. 1992: 16-17). The difference between Morpheme Properties and Allomorph Properties on the one hand, and "features" on the other, is that the former are visible to AMPLE constraints, while "features" are invisible to those constraints (and can therefore be used only in a program that uses AMPLE's output). It should be noted, however, that morphosyntactic features may be of great relevance to affixation processes; to the extent to which this is true, there will be duplication in an AMPLE analysis between the Morpheme Properties list and the "features" list.

Hermit Crab has two similar concepts, Rule Features and Head (morphosyntactic) Features. (Hermit Crab also implements Foot Features, but these are irrelevant to the discussion here.) Hermit Crab's Rule Features correspond to AMPLE’s Allomorph and Morpheme Properties,
while Head Features correspond approximately to AMPLE's "features." However, Hermit Crab morphological and phonological rules can be sensitive to Head Features as well as to Rule Features. This is not a problem in translating an AMPLE analysis into a Hermit Crab analysis, since nothing will be lost in the transfer (although it may be desirable to 'clean up' any duplication arising from the similar functions of AMPLE's Allomorph or Morpheme Properties and "features").

However, while there is little difference in the intuitive meaning of Morpheme Properties and "features" on the one hand, and Rule Features and Head Features on the other, there is a major difference in what the two programs do with these objects; this is discussed in section 4 below.

## 2.6 Morpheme Co-occurrence Constraints

Hermit Crab has nothing corresponding directly to AMPLE's morpheme co-occurrence constraints. However, a similar behavior can usually be imposed by the use of diacritic features. Specifically, a diacritic feature may be assigned to a morpheme whose appearance is required (or to a set of morphemes, any one of which satisfies the requirement); affixes which require the presence of that morpheme can require that such a feature be present on the stem to which they attach (or possibly on adjacent morphemes, as discussed in the preceding section).

As discussed in appendix 1, Hermit Crab also allows specifying that a given diacritic feature be absent, corresponding to a negative cooccurrence constraint.

## 2.7 Don't Load

This has no direct translation into Hermit Crab's model. Filtering out certain lexical entries is more properly the job of the interface; that is, lexical entries which are not to be loaded into Hermit Crab should not be passed to it in the first place. (LinguaLinks, for instance, has a concept of "filters" which could be adapted for this purpose.)

## 2.8 Comments

Comments have no purpose in Hermit Crab. Like lexical entries which are not to be loaded, comments can be stored in the dictionary database maintained by the user interface, but not passed to Hermit Crab.

## 3. Affix Dictionaries

As discussed above, AMPLE takes the IA perspective, and therefore considers affixes to be lexical entries (items) in dictionaries. Hermit Crab, on the other hand, takes an IP perspective in which affixes are treated as morphological rules which apply to derive one lexical entry from another. Nevertheless, there is a fairly direct translation from AMPLE's approach to Hermit Crab's (as argued for in the abstract in section 2 of the body of this paper). I base the discussion here on that in WBM chapter 11, section 2. The fields listed there, and their translation into Hermit Crab's conceptual view, are as follows:
3.1 Affix

AMPLE uses the affix marker only to define the beginning of the lexical entry record. As such, it is superfluous for Hermit Crab.

3.2 Allomorph

Inasmuch as AMPLE models MUF morphology, while Hermit Crab models SUF morphology, the allomorph fields represent a divergence between the two programs.

Inward sensitivity, that is, selection of allomorphs based on phonological or morphosyntactic properties of the stem to which an affix attaches, can be modeled in Hermit Crab by the use of subrules of morphological rules. For instance, the allomorphy of the English plural noun suffix could be represented as follows in Hermit Crab (as in appendix 1, I have used a notation that is likely to be familiar to linguists; the notation can be mechanically translated into the notation which Hermit Crab expects):

\[
\begin{align*}
[X_1 [+\text{student}]]_N &\Rightarrow [1 \ 2 \ \text{ez}] \\
[X_1 [+\text{voiced}]]_N &\Rightarrow [1 \ 2 \ z] \\
[X_1]_N &\Rightarrow [1 \ z]
\end{align*}
\]

(The '1' and '2' in the output of the subrules correspond to the first and second elements of the inputs, which are simply copied over to the output; see the discussion in appendix 1.)

Outward sensitivity in allomorphy requires the use of allomorphy rules, applied after all affixation is complete (or at least after all the affixes of a given stratum have been attached). It is of course quite possible to use allomorphy rules to describe inward sensitivity as well. Indeed, it might be more appropriate to use allomorphy rules for both inward and outward sensitivity, reserving the use of Hermit Crab subrules for other purposes. The adequacy of allomorphy rules for this purpose depends on their power, as opposed to the power of morpheme and string conditions in AMPLE. AMPLE's string conditions are essentially equivalent to the phonetic environment conditions of allomorphy rules in Hermit Crab. There are, on the other hand, significant differences between the treatment of Morpheme Properties in AMPLE and morphosyntactic features in Hermit Crab; see the discussion in section 2.5 above and in section 4 below.

3.3 Morph name

This corresponds directly to Hermit Crab's notion of a gloss. However, AMPLE allows the use of morph names in conditioning environments for allomorphs and in tests; Hermit Crab does not. There may be an argument for allowing such reference, or at least reference to diacritic features which would be carried by specific morphemes (which would amount to the same thing), and then referred to by allomorphy rules (see section 3.3 of the body of this paper).

3.4 Category pairs
AMPLE and Hermit Crab diverge in two ways in their treatment of categories (parts of speech). The first is a trivial syntactic difference: whereas AMPLE treats category pairs as single objects (along the lines of categorial grammar), Hermit Crab treats the input and output parts of speech of a morphological rule as separate objects. The second difference is akin to the difference between AMPLE and Hermit Crab with respect to the categories of stems: whereas AMPLE allows an affix to have multiple category pairs, an affix rule in Hermit Crab has a single input part of speech and a single output part of speech. As in the case of stem categories, the translation can be made trivially, by translating a single AMPLE affix into multiple Hermit Crab affix rules, one for each category pair. However, the result is liable to be very messy if the affix attaches to multiple categories. The extent to which this is true is likely to depend on the extent to which the linguist has used categories to make minor distinctions, rather than using morpheme properties. Another ameliorating factor is that Hermit Crab allows affixes to select for multiple subcategorizations. (Minor category distinctions in AMPLE, such as that between transitive and intransitive verbs, will often translate into distinctions in subcategorization in Hermit Crab.) Thus, the need for multiple Hermit Crab affix rules in place of a single AMPLE affix can be decreased by the use of multiple subcategorizations in place of multiple categories.

3.5 Order class

Hermit Crab does not directly support order classes. In many cases, order classes can be simulated by placing the affix rules in a linear order (as advocated e.g. by Anderson 1992). This may not be sufficient, however, if there are affixes whose appearance is unconstrained by the order classes.

Order classes can be simulated in Hermit Crab by the use of features. For instance, an affix which belongs to a particular order, say order 3, might bear the feature [3 order], and require that the stem to which it attaches bear the feature [2 order]. (If suffixes of the second order were optional, the third order affix could require the stem to bear either the feature [2 order] or [1 order], etc. Alternatively, third order affixes could prohibit the stem from bearing the feature [3 order] or any higher feature values.) The resulting stem would bear the feature [3 order], since the feature values of the affix override any conflicting feature values on the stem to which they attach. An affix whose appearance with respect to the order classes was free would place no restrictions on the order of the stem to which it attaches, nor would it bear any value for the feature [order] (rendering unnecessary the pseudo-order 'zero' and the associated test suggested in WBM section 11.2.5 (page 118) and section 13.4.1 (page 170f)).

3.6 Morpheme properties

Morpheme Properties and "features" of affixes behave identically to those of roots, so far as the built-in behavior of AMPLE is concerned. A comparison of this behavior with that of features in Hermit Crab appears in section 2.5 above, and section 4 below.

3.7 Morpheme Co-occurrence Constraints

AMPLE's Morpheme Co-occurrence Constraints do not have any direct translation in Hermit Crab, although they can generally be represented by a different concept. There are several cases,
depending on whether the constraints work in an inward or outward direction, and on whether they are positive or negative constraints.

An inwardly sensitive positive (negative) Morpheme Co-occurrence Constraint represents the fact that an affix requires (forbids) some other morpheme to be present in the stem to which it attaches. (Examples of both positive and negative constraints are given in WBM section 11.3.6, page 75.) A positive inwardly sensitive constraint can be modeled in Hermit Crab by assigning a feature value to the morpheme whose presence is required, and assigning a feature value requirement to the inwardly sensitive affix. A negative inwardly sensitive constraint can be modeled in the same way, except that the designated feature value(s) must be absent from the stem.

A Morpheme Co-occurrence Constraint that works in the outward direction, that is where a morpheme sets up a requirement that an affix attached outside of the given affix must be present, is represented differently in Hermit Crab. Consider for example the outwardly sensitive co-occurrence constraint discussed in WBM section 11.3.6 (page 129). This constraint says that a particular root must be followed by one of the suffixes whose names are 'IN', 'OUT', 'UP', or 'DOWN'. One way to recast this into Hermit Crab's model is to say that the suffixes in question bear a Head Feature [IN locative], [OUT locative] etc. (or perhaps [IN directional] etc.), and to mark roots which require these suffixes as having an Obligatory Head Feature of [locative] (or [directional]). The Obligatory Head Feature requirement means that some value of the designated Head Feature must be assigned by the end of the derivation. Since all non-locative (or non-directional) suffixes would lack a value for that feature, the effect would be to require the presence of one of the designated suffixes. Outwardly sensitive constraints set up by affixes will work in the same fashion, since a morphological rule can add Obligatory Head Feature requirements.[21]

3.8 Infix location

'Infix locations' in AMPLE have two uses:

1. to define the sort of morpheme into which another morpheme can be infixed; and
2. to define the phonological environment in which the infix may be found (after the first consonant of a stem, etc.).

In Hermit Crab, (1) may be treated by ordering the infixation rule with respect to other affixes or by the use of order features (see section 3.5 above), and (2) is directly implemented as part of the morphological rule for the infix using a notation not unlike the AMPLE notation (with the usual caveat that AMPLE deals with string classes, while Hermit Crab deals with phonetic features specified by natural classes and/or phonemes).

3.9 Don't load

As with lexical items which represent stems (see section 2.7 above), there is no direct translation of this property into Hermit Crab's model. Filtering out affixes which are not to be loaded is the job of the user interface.
3.10 Comment

Again, comments should be treated by the user interface, and not passed to Hermit Crab at all (see section 2.8 above).

4. Tests

In addition to assigning string constraints and morpheme constraints to allomorphs, AMPLE allows the user to set up more general tests to be applied to analyses (WBM, chapter 13). AMPLE’s tests in effect constitute a programming language approach to morphology, as opposed to a linguistically based approach (cf. Sproat 1992, section 3.6.5, particularly footnote 62). For instance, AMPLE has no built-in behavior with regard to Morpheme Properties; any special behavior must be programmed in by the user. Nor does AMPLE have any built-in notion of a hierarchy of morphological structure; one can only write tests concerning morphemes to the left or right of a given morpheme. Hermit Crab, on the other hand, takes a linguistically-based view of features in which features are percolated up from an affix to the next level in a word's hierarchical structure, along with any nonconflicting features of the stem to which the affix is attached (see appendix 1). An affix may require (subcategorize) the presence or absence of certain feature values on the stem to which it attaches, but there is no notion of searching the morphemes to the left or right for those feature values.

Because of AMPLE’s programming language approach, there are a great many tests that one could perform in AMPLE which have no direct equivalent in Hermit Crab. It is not my intention to explore all the possibilities AMPLE’s tests offer, as its programming language approach would render this an open ended exercise. Rather, I will investigate a few cases where there is an approximate correspondence between tests in AMPLE and Hermit Crab tests, on the assumption that these are the sorts of tests which appear to be more linguistically motivated and therefore more likely to be used in actual analyses.

4.1 fromcategory, tocategory

AMPLE does not automatically impose category restrictions on the attachment of affixes; the relevant tests must be written by the user. Some common tests are:

    right tocategory is current fromcategory

for prefixes,[22] and

    left tocategory is current fromcategory

for suffixes.

However, things are considerably more complicated if the layering is complex, for instance if the language has both prefixes and suffixes, particularly if the 'layers' of the word alternate between prefixes and suffixes. The question of finding the 'FINAL category' in AMPLE is also complicated if there are both prefixes and suffixes. In Hermit Crab, on the other hand,
propagation of categories is built in, in the sense that the part of speech after attachment of an affix will be the part of speech of the affix, if this is defined, or else the part of speech of the stem to which the affix is attached. Thus, the propagation of parts of speech is determined directly by the hierarchical structure of the word, regardless of the left-to-right order of affixes. The outermost affix for which a part of speech is defined determines the part of speech of the final word. (In the case of compounds, the part of speech is determined in the compound formation rule by which constituent is explicitly marked as the head.)

It is also common in AMPLE to test that the 'FINAL tocategory' belongs to a set of admissible categories, in order to ensure that all necessary affixes have been attached. Hermit Crab takes a different approach to ensuring completeness of affixation: morphemes (both roots and affixes) can specify a set of 'obligatory features' (i.e. morphosyntactic features, such as person or tense); if any obligatory feature does not receive a value by the end of the derivation, the derivation is ruled out.

In general, many of the tests done in AMPLE with categories (parts of speech) are more conveniently treated in Hermit Crab with features or subcategorization lists. Consider for instance a causative affix, which allows a verb to take an additional grammatical object. In AMPLE, such an affix might be specified to have a large number of part of speech pairs: one for each subcategory of verb (intransitive, transitive, ditransitive, citative, particle-taking verbs, etc.). In Hermit Crab, it would be possible to take a more modern approach by assigning the part of speech "Verb" to all verbs, distinguishing the various subcategories by means of the lists of complements each takes (no complements for intransitive verbs, a single NP with appropriate case marking for transitive verbs, etc.). The causative affix would then make no change to the part of speech, instead appending a single NP to the verb's complement list. Thus, instead of the long list of part of speech pairs required in AMPLE, in Hermit Crab the causative affix would have a single input part of speech requirement and effect a single change to the complement list of the output. Moreover, adding a word belonging to a new subcategory of verbs to AMPLE would imply not only the addition of the word to the lexicon, but also a concomitant change to the part of speech pairs of the causative affix (and any other affixes which attach to verbs). In contrast, the addition of a word belonging to a new subcategory of verbs in Hermit Crab would require only the addition of the new word (including its complement list) to the lexicon; no change whatsoever would be required to the morphology.

4.2 orderclass

See the discussion above (section 3.5) concerning order classes.

4.3 left/right property

In AMPLE it is possible to test for morpheme properties on the adjacent morpheme on the left or right using 'left property' or 'right property' (equivalent MECs can be written). Such a test is not possible in Hermit Crab, as this program's behavior is currently specified. As suggested above (section 2.2 of this appendix), this is arguably a shortcoming of Hermit Crab's current specification, and could easily be remedied.
4.4 FOR ALL LEFT/RIGHT, FOR SOME LEFT/RIGHT

In AMPLE one can require that *all* morphemes to the left or right (or both) of a given morpheme bear some property, or not bear some property, using the FOR_ALL_LEFT or FOR_ALL_RIGHT operators; or that there be *some* morpheme to the left or right which bears or does not bear some property, using the FOR_SOME_LEFT or FOR_SOME_RIGHT operators. A test requiring that some morpheme to the left or right bear a particular Morpheme Property is the equivalent of a Morpheme Environment Condition (MEC) requiring that Morpheme Property to appear to the left or right. The advantage of using the test, rather than a MEC, is that the test can be written once, and used for a number of morphemes. In the same way, a test requiring that no morpheme to the left or right bear a Morpheme Property is equivalent to an MEC barring the Morpheme Property.[24]

Hermit Crab has rough equivalents to these two tests (or their equivalent MECs), in that an affix can require that the stem to which it is attached bear a particular feature value or not bear a feature value. This is approximately equivalent to a FOR_SOME_LEFT/RIGHT test because of the percolation conventions built into Hermit Crab. [25] However, while the intent of such a test in AMPLE is often close to the intent of the test in Hermit Crab, there are a number of asymmetries between percolation on the one hand, and left/right searches on the other. Thus, an AMPLE test might find a morpheme with the appropriate property that is attached 'outside' the affix in question, rather than 'inside' (i.e. in the stem to which the affix attaches), whereas Hermit Crab can only test for morphosyntactic features percolated from the root or from affixes attached inside a given affix. One situation in which the AMPLE test might be more appropriate would be that of extended exponence, where the principal exponent of the feature in question is an outer affix. In such cases (which, according to Carstairs 1987, are rare), the outward sensitivity will need to be handled by an allomorphy rule applying after the outer suffix is attached, assuming the test in question determines the choice of allomorph.

There is also the possibility in Hermit Crab that a morpheme 'inside' the given affix does bear the feature value in question, but that the stem to which the affix attaches does not bear a feature with the same value because another affix with a different value for that feature was attached later. A diagram may make this clearer:

\[
\begin{bmatrix}
\text{root} \\
\vdots \\
\text{affix1} \\
\text{affix2}
\end{bmatrix}
\]

In this hypothetical word, \textit{affix1}'s value for the feature \textit{foo} 'shields' \textit{affix2} from seeing the root's value for that same feature. Whereas in AMPLE, one could assign a test for the presence or absence of the feature [+foo] as a condition on the attachment of \textit{affix2}, regardless of the presence of the property [-foo], in Hermit Crab it is only possible to test for the outermost value of that feature, in this case its value on \textit{affix1}. Is this restriction in Hermit Crab a great loss? I would doubt it; it is, after all, predicted by the linguistic theories on which Hermit Crab's behavior is based, that there will be no need to test the value of such a 'shielded' feature. Putting this differently, the behavior of AMPLE in this situation is linguistically inappropriate.
Note that in Hermit Crab, it is possible for affix1 to be a suffix and affix2 a prefix; what is relevant is not the linear ordering of the morphemes, as in AMPLE, but their hierarchical relationships. There is no corresponding notion of hierarchy in AMPLE. Again, the behavior of Hermit Crab is based on linguistic principles, and is therefore preferable.\textsuperscript{[26]}

Similar to the above test would be a requirement in Hermit Crab that a stem not bear any value for a given feature (implemented using a designated null feature value), corresponding roughly to the AMPLE requirement that no morpheme bear a particular property. In this case, the correspondence between AMPLE and Hermit Crab is closer because it is not possible for an intermediate affix to 'shield' a nonexistent feature value.

Apart from the question of shielding of one value of a feature by another value of the same feature, and the prefix-suffix distinction, then, there is a rough correspondence between an AMPLE test of the form 'for some morpheme to my left (right), property X holds' and the Hermit Crab requirement that a stem bear feature value X. Likewise, there is a rough correspondence between an AMPLE test of the form 'for all morphemes to my left (right), property X does not hold', and the Hermit Crab requirement that a stem not bear feature value X. AMPLE also allows tests of the form 'for some morpheme to my left (right), property X does not hold' or 'for all morphemes to my left (right), property X holds.' There are no corresponding tests in Hermit Crab.\textsuperscript{[27]} But again, it seems unlikely that these tests represent a correct view of what happens in natural language, so we may not be missing out on much by not allowing for such tests in Hermit Crab.\textsuperscript{[28]}

**Appendix 3: How Are Allomorphy Rules Applied?**

1. **Introduction**

This appendix discusses several more subtle points concerning the application of allomorphy rules, namely:

   1. The ordering of allomorphy rules among themselves;
   2. The interaction of allomorphy rules with phonological rules; and
   3. The interaction of allomorphy rules with morphological rules.

2. **Ordering of Allomorphy Rules**

In classical generative phonology, the answer to the question of how allomorphy rules were ordered among themselves was straightforward: readjustment rules, of which allomorphy rules were a subclass, were linearly ordered. But in Multiple Underlying Form (MUF) theories, conditions on the appearance of allomorphs in MUF morphology are generally construed as conditions to be met at the surface, i.e. as surface-true generalizations. If this is what is desired, linear ordering is the wrong way to apply allomorphy rules; in fact, simultaneous ordering would be incorrect too, as that would result in the allomorphy being conditioned by the underlying forms, not the surface forms, leaving open the possibility of non-surface-true behavior.
One way to apply allomorphy rules in a surface-true fashion would be to treat the application of allomorphy rules as constraints, rather than as rules. The application of a set of allomorphy rules then constitutes a constraint satisfaction problem. Suppose that for each morpheme, there is an allomorphy rule, with each allomorphy rule having one or more subrules. The subrules are ordered from specific to general, so that if two subrules are applicable to the same representation, only the more specific subrule applies. (A morpheme with no allomorphs has a single subrule, an 'elsewhere' case that applies in any environment.) The problem then becomes one of applying the allomorphy rules to a form in such a way that one subrule of each rule is satisfied; and for each such subrule which has applied, no more specific subrule would be applicable. The result is that each morpheme's allomorphy rules are satisfied in the output, i.e. the rules describe true generalizations at the point at which all allomorphy rules have been applied.

3. Interaction of Allomorphy Rules with Phonological Rules

In classical generative phonology, the question of when in the course of a derivation allomorphy rules were to be applied also had a simple answer: the allomorphy rules applied in a block after affixation, and immediately before the true phonological rules.[29] This also works in an MUF system in which there are no (morpho-)phonological rules. It is not clear that it will always produce the desired results in a mixed system in which some allomorphs are generated from underlying forms by phonological rule, and some by allomorphy rules, if the allomorphy rules are intended to represent surface-true generalizations. (Such a situation might arise in the process of gradually converting an MUF analysis into an SUF analysis.) That is, an allomorphy rule may cease to describe a surface-true generalization because the phonological rules may, after the allomorphy rules have applied, alter either the affixes to which the allomorphy rules have applied, or parts of the word which represented the environment in which those allomorphy rules were applied. The application of the allomorphy rules might be revised so that their structural changes would be applied before the phonological rules applied, but their environments checked afterwards; but there would be a large price to pay in terms of efficiency, and it is not clear that it would be justified. In general, it seems better to apply the allomorphy rules before the phonological rules, with the understanding that phonological rules may further alter the output of the allomorphy rules.

A difficulty could arise if a phonological rule obscured the boundary between two morphemes, making it difficult to determine where a morpheme ends and its environment begins. If, as I have advocated, allomorphy rules apply in a block before true phonological rules, this situation could arise only under cyclic rule application. Furthermore, while phonological rules could obscure the boundaries under autosegmental phonology, they are less of a problem with classical (segmental) generative phonology. Parts of a morpheme may disappear (by deletion rules), and non-morphemic material may be inserted (by epenthesis rules), but two morphemes may not be intermingled unless the phonology includes metathesis rules. Nonetheless, if the morpheme has been modified by phonological rules on one cycle, it may no longer match against the structural description of an allomorphy rule on a later cycle. It is unclear to me what the proper behavior should be in this case.

4. Interaction of Allomorphy Rules with Morphological Rules
Another difficulty arises in the application of allomorphy rules if there is nonconcatenative morphology: the morpheme to which the allomorphy rule is to apply may not be identifiable, or it may not be composed of a contiguous set of segments (phonemes). Some examples may make this clearer. Consider first the morphological process of complete reduplication. After a word has been reduplicated, it is no longer evident which part of the resulting word is the original stem and which is the 'affix' of reduplication. If the stem is composed of morphemes which have allomorphs, it is therefore unclear to which portion the allomorphy rules should apply.

Infixed may result in a similar problem. In the case of the infix itself, the difficulty is probably not too great; if we are dealing with a single contiguous infix, its left and right environment begin at its left and right end. However, if the infix has more than one part (e.g. Semitic vowelining, under one analysis), then what counts for the infixes' left and right environment is no longer apparent. Furthermore, the stem into which an infix is inserted is broken up by that infix, meaning that its constituent morphemes may have been broken up. Indeed the breaking up of a morpheme by an infix is the common situation, without which there would be little reason to consider the affix to be an infix, rather than a prefix or suffix. The result is that the domain of the allomorphy rule for the morpheme into which the infix has been inserted is no longer contiguous; it may be necessary to treat the infix as invisible for the purposes of the allomorphy rule affecting the morphemes composing the stem.\[30\] (This is not unlike the multiple tier analysis of Semitic morphology proposed by McCarthy 1979.)

Null affixes are also problematical, since it is arbitrary whether they should be prefixed or suffixed to a stem, rendering the question of left vs. right environment problematical. (We are not concerned so much with multiple null allomorphs of a single morpheme--an unlikely analysis--as with the question of an affix with non-null allomorphs in some environment, and a null allomorph in some other environment.) One solution would be to arbitrarily treat null affixes as prefixes or as suffixes; a better solution might be to disallow null affixes as underlying forms if there are non-null allomorphs.

Subtractive (truncation) affixes present an even worse problem of distinguishing stem from affix; fortunately, subtractive affixes are rare almost to the point of nonexistence.

Simulfixes also cause problems in the application of allomorphy rules. Consider the morphological process which applies to some stop-initial stems in Mezquital Otomi (Wallis 1956) by voicing the initial stop; the following rule might be used to express this process:

\[
\begin{bmatrix}
C \\
1 \\
2
\end{bmatrix} \rightarrow \begin{bmatrix}
1 \\
\text{[+voice]}
\end{bmatrix}^2
\]

After the above rule has been applied, it is no longer clear where the environment ends and the stem begins: does the stem include the feature [+voice], or is that part of its environment?

Similarly, circumfixes make the left and right environments ambiguous.

In general, it will be difficult to establish a boundary between stem and affix whenever the affixal process has modified the stem by doing anything except attaching phonological material
either before or after the stem (but not both), or inserting more than a single infix. Note that prefixation and suffixation, together with simple infixation, are the classical cases where IA morphology works well; when allomorphy rules are used with 'ordinary' IA morphology, there is no difficulty in distinguishing affix or stem from environment. I have no definite solutions to offer in this case, save to say that an IP analysis in both senses of the term (affixes as rules, and single underlying forms) may be more appropriate in these cases.

5. Conclusion

A mixed system, one with both phonologically conditioned allomorphs and phonological (morphophonemic) rules is not a 'theoretically pure' system, and it is difficult to know what its proper behavior should be. Mixed systems are quite likely to arise in practice for a field linguist, no matter what his commitment to SUF morphology (as argued in section 3.2 of the main text). I have argued that the proper behavior is:

1. Allomorphy rules should express true generalizations at the point at which the block of relevant allomorphy rules has finished applying; in other words, they should be viewed as constraints rather than as rules.
2. Allomorphy rules should be applied in a block before (true) phonological rules are applied. There are potential problems with cyclic phonology, for which I have no clear answer; fortunately, such problems appear to be minor within the context of segmental (as opposed to autosegmental) phonology.
3. There are some difficulties with regard to nonconcatenative morphology and cyclic phonology, but these appear to be unavoidable (and represent difficulties for IA morphology in any case).

Endnotes

* I have benefited from comments on a draft of this paper by Albert Bickford, Andy Black, Bill Mann, and Gary Simons.

1 Hockett also mentioned a third classification for theories which make essential use of paradigms; I will have nothing to say about such theories in this paper, although I believe the issues for computational implementation are much the same.

2 Terminological issues can cause confusion here; "lexicon" is often used to refer not only to listed morphemes--what Di Sciullo and Williams (1987) refer to as "listemes"--but to the entire set of derived and even inflected words, including forms derived by rule. I will use the term "lexicon" in the more restricted sense of the set of listemes. I will also ignore the question of the lexical representation of idioms and other phrasal categories.

3 I ignore the issue of whether the rule is triggered by the presence of the plural feature on the noun, or whether the rule itself adds the plural feature. Among those working in the general framework of IP morphology, the former is the position taken by those working in a Word-and-Paradigm model, such as Zwicky (1985) and Anderson (1992), while the latter is the position of Halle and Marantz (1993).
Again, I ignore the issue of whether the lexical entry carries a plural feature which is percolated upwards, or whether it subcategorizes for an existing plural feature. The former is roughly the position of Lieber (1980) and of Di Sciullo and Williams (1987), while the latter is the position of Matthews (1972a) and Matthews (1972b), if his work is interpreted as Item-and-Arrangement.

The modifications to IA morphology needed in order to circumvent the traditional arguments against this approach are by no means minor, and it is not clear that they can be translated into a segment-based view of phonology. For instance, the attachment of affixes under extrametricality (McCarthy and Prince 1990) would be problematical.

I am not making the converse claim, that all allomorphy rules can be mechanically translated into multiple allomorphs. The translation from multiple allomorph statements to allomorphy rules suffices for our purposes.

The question or ordering arises when allomorph rules are used to encode allomorphy; I return to this issue in appendix 3.

The reverse translation, from an SUF analysis to an MUF analysis, might also be possible. A problem could arise translating allomorphy rules that were to apply to more than one morpheme, however.

Perhaps Haeckel was right; perhaps ontogeny does recapitulate phylogeny.

Admittedly, this is not always possible, for instance if the description uses mechanisms of IP morphology which have no counterpoint in IA morphology.

Relaxing the minimum length from one to zero gives phonologically null stems, should this be deemed desirable.

Familiar to linguists, at least. The effect is not unlike a case structure in many programming languages, with the code to be executed by each case statement followed by a break, and the final case applying if none of the earlier cases have applied.

This is basically the notion of feature percolation advocated by Lieber (1980) and Di Sciullo and Williams (1987). Realizational morphology, as advocated by Matthews (1972a), Matthews (1972b), Zwicky (1985), and Anderson (1992), might be treated under Hermit Crab as rules which place requirements on the morphosyntactic features of the stem to which they attach, but do not provide any features of their own. A convention for specification of the features to be realized on a given word would need to be added to Hermit Crab.

More accurately, it will. This behavior is not implemented yet.

A common use of negative string environment constraints is for phenomena which are dependent on position within a syllable. For instance, if an allomorph appears in syllable-final position, it may be easier to say "not before a vowel" than "before a consonant or word-finally" (given that syllable structure itself is not usually represented in the strings which are the input to
parsing). One solution available in Hermit Crab would be to encode syllable structure in (pseudo-)phonetic features, assign those features to a lexical representation by rule, then require that the allomorph appear syllable-finally.

16 In the case where allomorphs are treated by the use of multiple subrules of morphological rules, Hermit Crab can assign particular Morphological/Phonological Rule Features or Head (morphosyntactic) Features to allomorphs. However, the use of morphological subrules is arguably not the best way to treat allomorphy.

17 If there is no generalization to be captured -- i.e. if only one or two morphemes undergo the process -- then the appropriate descriptive technique is to use allomorphy statements local to the morphemes in question, using MECs (in AMPLE) or allomorphy rules (in Hermit Crab).

18 It is unclear in this particular example whether the input to the rule can be stated in purely phonological terms, or whether it requires diacritic (rule) features. The environment will require either diacritic features, or a more abstract representation of certain morphemes, because while the shortening usually applies in a particular phonetic environment (before a consonant cluster), it also applies before certain suffixes which synchronically do not contain consonant clusters. In the more general case, of course, allomorphy is a result of general phonological processes.

19 Focusing on a different aspect of the problem, some linguists might argue that the AMPLE approach is better, in that it uses a constraint-based approach instead of a rule-based approach. This issue is, however, orthogonal to the one in the text, inasmuch as constraint-based approaches (such as that in Bird 1995) are like rule-based approaches in that they describe generalizations in terms of phonological structure, rather than in terms of allomorph properties.

20 Compare the warning against relying too heavily on order classes in WBM, section 4.4 (page 35).

21 A realizational morphology approach would also be possible, and perhaps superior; cf. footnote.

22 Actually, the test for prefixes would probably be somewhat more complicated than this, due to AMPLE's left-to-right processing (the test given in the text could only be done as a final test). I abstract away from this issue.

23 Adding a filter to Hermit Crab to block analyses in which the outermost part of speech is not a member of a specified set would be trivial. The use of Realizational Morphology would also be a way of ensuring completeness of affixation.

24 Tests can also be written requiring that all morphemes to the left or right bear a feature, or that some morpheme to the left or right not bear a certain feature; such tests are of lesser utility, as I will discuss below.

25 Here I abstract away from the distinction in AMPLE between final tests and tests on the current morpheme.
26 It has been argued that prefixes and suffixes behave differently with respect to feature percolation (Di Sciullo and Williams 1987), but this is far from universally accepted. This difference could be built into Hermit Crab, if necessary.

27 More (or perhaps less) perspicuously, the following two AMPLE conditions have rough correlates in Hermit Crab (abstracting away from left/right conditions):

\[
\begin{align*}
\exists m \ X(m) \\
\forall m \ (\neg X(m))
\end{align*}
\]

But the following AMPLE conditions do not have correlates in Hermit Crab:

\[
\begin{align*}
\exists m \ (\neg X(m)) \\
\forall m \ X(m)
\end{align*}
\]

28 It might seem that the notion of conjugation classes (or declension classes) would require a condition of the form 'for every morpheme X, class(X) holds'. But in fact, membership in a given class is typically a property of a root (or of a derived stem), and the affixes used by a given class impose a requirement on the stem to which they attach that it belong to a particular class. Thus, inflectional affixes do not bear a feature for conjugation class, they only check it. The check is easily implemented in Hermit Crab as a required feature constraint on the affixes affected.

29 Note that under cyclic application, an allomorphy rule may apply to an affix on a cycle later than that on which the affix is attached. An example is the allomorphy rule governing the suffix -fy (example () in the text).

30 Such an analysis would imply that it would be impossible to have allomorphs of the stem which are conditioned by the shape of the infix.

References


