Morphology as a Computational Problem

UCLA Occasional Papers #7:
Working Papers in Morphology

Edited by Karen Wallace

Department of Linguistics
University of California, Los Angeles

January 1988

Contents

Introduction

This collection of papers is an outgrowth of a proseminar on parsing morphology which took place at UCLA in the winter and spring quarters of 1985. This was a new area of interest for most of us, and an especially exciting one for those who had studied languages in which the productive morphology is complex enough to justify creating computer programs to do word analysis. Steve Anderson's contribution to this volume (Chapter 1) discusses some of the reasons for doing morphological parsing, and explores theoretical and computational issues encountered in the parsing of complex morphologies.

We were fortunate during this time to have access to an early implementation of *keçi* (known to us then as TPARSE), Jorge Hankamer's parser for Turkish words; Jorge came and gave a talk explaining how it worked (see Hankamer (1986)). His paper in this volume (Chapter 7) discusses ways that his parser, originally designed only to handle affixation, could be modified in order to correctly account for nominal compounds in Turkish.

In addition to *keçi*, we had a working version of *KIMMO* (Karttunen (1983), Koskenniemi (1984)), converted to PROLOG code by Sean Boisen. His description of this task, complete with the lexicon and automata files for English, appears in this volume as Chapter 3. It includes discussion of the two-level model and the advantanges and disadvantages of Prolog in its implementation.

In the course of working on projects with these parsers and the morphology and phonology of various languages, many of us were inspired to write our own parsers: parsers that would overcome the problems we found, parsers that would be elegant and general and theoretically satisfying, parsers that would incorporate autosegmental phonology and embody a-morphous morphology. We would create the Parser to End All Parsers; the cockroach that ate Cincinnati would not hold a candle to the parser that ate *mikhuykakaporqanchejpuni*. The 'Definite Clauses' paper by Sean Boisen (Chapter 2) and the papers by Tom Cornell, Bill Dolan, and Karen Wallace (Chapters 4, 5, and 8) represent 'progress reports' of our work along

this dimension.

Finally, on the psycholinguistic side of morphological parsing, Karen Emmorey's contribution (chapter 6) presents the psychological evidence that humans parse words, and discusses models of parsing that might account for the way in which they do it.

This research was supported in part by grant number BNS-84-18277 from the National Science Foundation. The editor is grateful to Steve Anderson for his help with LaTeX .

Chapter 1

Morphology as a Parsing Problem

Stephen R. Anderson

The problem of analyzing the structure of words computationally in Natural Language Processing systems has traditionally been solved by 'throwing silicon at it'.[1] If one simply lists all of the words that might be encountered (possibly abstracting away from a limited number of affixes which are stripped off in a preliminary step), together with an indication of their relevant properties, the analysis can get on with the other sorts of task to which real NLP systems are dedicated (syntactic parsing, textual searches, etc.). The practicality of this approach, however, is in large part an artifact of the properties of English, the language to which by far the majority of NLP work has been devoted. Since English has very little inflectional morphology, and its derivational morphology is often associated with lexical idiosyncrasy, the ratio of possible word forms encountered in text to necessarily listed lexical items is comparatively low.

When one attempts to generalize this technique to other languages, however, it

[1]This research was supported in part by grant number BNS84-18277 from the US National Science Foundation. I would like to thank various people for comments and suggestions on the problems discussed here, including Alfonso Caramazza, Tom Cornell, Bill Dolan, Karen Emmorey, William Marslen-Wilson, Karen Wallace, Eric Wehrli, and Amy Weinberg. As usual, none of these people should be blamed for my mistakes or misconceptions. Special thanks are due to Mark Seidenberg, who has repeatedly pushed me to make it clearer that there is some principled reason for pursuing these problems. I doubt that I have really satisfied him, but I am much clearer in my own mind about why computational morphology is interesting than I would have been without the stimulus of his questioning.

becomes clear that it is both costly and linguistically inadequate. In languages like Finnish or Georgian, for example, each lexical verb may have literally thousands of distinct surface forms, differing in inflectional properties but otherwise the same vocabulary element. Of course, this does not by itself make it impossible to list all of the forms: the numbers involved are after all finite, if large, and the price of memory seems to come down every year. Nonetheless, it seems wasteful to devote such vast resources to lexical listing, given the considerable degree of predictability of most surface word forms in such cases. More importantly, perhaps, the word space of a language is typically quite highly structured, and insofar as its dimensions are determined by linguistically significant regularities we *ought* to be able to extract the information borne by words on the basis of these regularities rather than simply listing them all. From a theoretical point of view, indeed, we have clearly failed to characterize the structure of the language insofar as this substantial area of regularity is treated implicitly as if every word were equally arbitrary.

On this basis, then, an interest in the computational analysis of morphology has awakened in recent years. However, when we ask in a computational system for an account of the information carried by individual word forms, there are two rather different sorts of motivation we might have, and the extent to which we want to pursue a linguistically well motivated analysis will depend on the goals of the analyst. An understanding of this difference, in turn, is important in determining the sorts of criticism of a given system that might be appropriate.

On the one hand, the analysis might be undertaken for essentially practical reasons: the question of what information a word form carries is then posed because we want the answer. For example, in developing a procedure for syntactic analysis, we generally need to start from an identification of the words in each sentence or text. If our interest is in the syntax, we may not care how this information about words is derived, so long as it is expeditiously made available. Another sort of practical motivation is for the purpose of identifying the lexical content of documents. Some work on document retrieval systems provides a rather precise procedure for searching a large database, founded on the construction of lexical profiles of each potentially relevant document. This method requires a technique for identifying which of the words in a document are the same and which are different — apparently a rather straightforward matter of string comparison for a language like English (again, making some provision for stripping away a small number of common inflectional affixes such as final *-(e)s*, *-(e)d*, etc.). Extending it to documents in languages with substantial morphology (such as Russian), however, necessitates a more principled way of abstracting away from morphological differences in the surface shapes of the 'same' lexical item.

When we approach morphological problems for purely applied reasons such as

these, the primary criterion of adequacy can reasonably be said to be efficiency, rather than theoretical motivation. If a system performs its task quickly and without complaint, we may not care whether it conforms to linguistic principles or generalizes to languages with very different structures. In contrast, we might ask the question of how morphological form can be related to content precisely for reasons of principle: because we want to understand the question, not because we have a use in mind for the answer. On this basis, the linguist interested in morphology from a theoretical point of view may well want to explore the implementation of a particular theory in a computational system because of the great increase in rigor and precision that such a formulation requires. For this purpose, the virtue of a computer is that it knows nothing it has not been told, and so when one reaches a point at which a computational procedure operates correctly, it is reasonably certain that all of the underlying assumptions and procedures have indeed been made fully explicit. In general theoretical discussion which proceeds without such constraints, a certain amount of handwaving is usually taken to be acceptable, and the promissory notes thereby issued are rarely called in. The requirements of specifying a computation, however, do not allow the theoretician to live on this sort of credit.

Perhaps a more interesting reason of principle for pursuing computational morphology is the possibility it offers of saying something serious and testable about linguistic processing. Linguists usually preface theoretical accounts with a disclaimer that these relate to competence, the structure of the knowledge which we attribute to a speaker, rather than to performance, the way that knowledge is actually put to use. This is surely a defensible, even essential idealization in identifying a coherent domain of study for an emerging science; but the question of specifying the relation between a grammar of competence and its use in a model of performance has thereby been only postponed, not avoided altogether.

Now unlike a grammar that simply specifies the properties of forms in a language, a computational system actually exhibits a sort of 'behavior': when presented with an input for analysis, it eventually emits some information after pursuing some determinate procedure in real time. The procedure in question is of course founded on some sort of rules that can be taken to specify the structure of the system's 'knowledge'. Insofar as we can establish a reasonable correspondence between the system's properties and actual speaker behavior, we can claim to have developed (at least part of) an account of this aspect of linguistic performance. Furthermore, to the extent that the structure of the system's 'knowledge' can be based on the principles of linguistic theory, we have validated the notion that knowledge of this form could indeed underlie such performance.

The advantage of morphology for such a demonstration is due to the existence of a vast literature in psychology on the fine structure of human behavior when

performing much the same task a computational system is engaged in: recognizing and retrieving words and recovering the information they bear. Lexical access and lexical decision tasks are a standard paradigm in psychology, and a great deal is known about how words of various types are recognized.[2] As a result, the implicit claims about performance made by a computational system are probably more significantly testable in the domain of morphology than elsewhere in the study of natural language. Computational morphology thus offers us a chance, at least in principle, to determine whether or not the constructs of (at least part of) linguistic theory are appropriate as a description of human linguistic behavior.

1.1 Approaches to Computational Morphology

The usual picture of word structure taught in elementary courses in linguistics suggests that words are made up of smaller pieces strung together — minimal signs or 'morphemes' that constitute the atoms both of word form and of word meaning. In that case, we might approach morphological analysis as the problem of chopping words up into their component morphemes so that these (presumably irreducible) elements can be individually looked up in a lexicon. In fact, substantial evidence exists that this picture is seriously oversimplified and that the notion of the morpheme impedes the understanding of morphology rather than helping it,[3] but even if we accept it the problem is quite non-trivial. Consider, for example, how we might go about analyzing the Finnish word *karahkoja* 'stick (partitive plural)'. This form is apparently composed of three elements, which could be assigned the lexical shapes /karahka/ 'stick', /i/ 'plural', and /ta/ 'partitive'. The concatenation of these elements is related to the surface form by the operation of several phonological rules of Finnish,[4] as in (1):

(1)	/karahka+i+ta/	'stick'+'plural'+'partitive'
	karahko+i+ta	(a → o before i)
	karahko+i+a	(t → ∅ between short vowels)
	[karahkoja]	(glide formation)

As is apparent, none of the components of this form (whose shape is justified by the phonology of the language and their appearance in other words) appears unaltered in its surface shape, due to the effects of the phonology. Before they

[2]See Emmorey (1987) for a review of much of this material.

[3]See Anderson (1987) for discussion of this issue and its place in a theory of morphology.

[4]Cf. Keyser and Kiparsky (1984) for discussion.

can be looked up in a dictionary of morphemes, then, it would be necessary to compensate for these obscuring influences.

The most straightforward way to resolve this difficulty would seem to be simply to 'undo' the phonological rules: applying them in reverse to the surface form, we should arrive back at the underlying shape. From there, in turn, we can attempt to recover the divisions between units by looking in the lexicon to see how the form can be exhaustively covered by lexical items (each representing one morpheme). This sort of approach can be called 'bottom up', since it starts exclusively from the given form and attempts to find a way of grouping its parts into larger structures.

The approach just outlined was proposed by Kay (1977) in one of the first works to take morphology seriously as a parsing problem. Kay proposes that the procedure for analyzing words is essentially as in (2):

(2) a. Starting from the surface form, undo the phonology by inverting the rules. Construct a set containing all of the phonologically possible underlying representations that could have given rise to the form being analyzed.

b. See if (at least one of) these representations could correspond to a well-formed sequence of lexically listed elements.

While it appears to be indicated directly as a matter of simple common sense, Kay's approach was never in fact developed for any language with a significant degree of phonological complexity. A major problem with this pure bottom-up approach arises directly if the phonology of the language involves a non trivial amount of neutralization (through assimilations, deletions, insertions, etc.). When that happens, it is necessary to calculate all of the possible combinations of alternatives allowed by various rules, which may be individually large when neutralizations are involved and whose product grows exponentially as the amount of significant rule interaction (ordering) increases. The combinatorial possibilities involved in undoing the phonology thus get out of hand rather quickly. Since the depth of ordering in a linguistically motivated description can easily approach 15–20, with many of the rules involved being many-ways ambiguous when regarded from the 'wrong end', the approach of simply undoing the effects of the rules was soon seen to be quite impractical.

An alternative to Kay's bottom up approach was developed by Jorge Hankamer for purposes which were originally purely practical. In entering a large corpus of Turkish text into a computer, Hankamer wanted to be able to check typographical accuracy. Naturally, this cannot be completely automated, but a good approximation would be to check that each word entered was in fact a possible word of Turkish. Given the extensive morphological apparatus of the language, however, this could clearly not be done by any sort of exhaustive listing: rather, Hankamer's

goal was to check acceptability by attempting to recognize each word. This clearly leads directly to the construction of a morphological parsing procedure.

The procedure Hankamer developed is based on a representation of the morphological system of Turkish (its 'morphotactics') in the form of a finite state machine. Without going into details,[5] this representation relies on the fact that, at any point in the analysis, the possibilities for the identity of the next piece of material in a word are severely constrained by the material that has already been identified. Nominal suffixes only go on nominal stems, for example, and the relative order of different suffixes is narrowly limited.

Hankamer's procedure exploits this as follows. The parse starts at the left edge of the word, since Turkish is an exclusively suffixing language and the lexical root is always found in this position. An attempt is made to identify as much as possible of the word with some lexical stem. Once a stem is found, and assuming that it does not exhaust the word, the finite state machine representing the morphology is consulted to see what material could come next. Given the language's system, the number of following morphemes is quite limited; each of these is tried in succession, by attaching it to the stem and then applying the phonological rules of the language to the result. Insofar as the result is consistent with the form being analyzed, this is taken as confirming the possibility that the element just added might be part of the parse. Once again, if the word has been exhausted (and the finite state machine is in a possible final state), the accumulated parse is reported as a result; otherwise, the next step in the finite state machine is tried and the process continues to completion (or failure). At any point, when none of the morphologically acceptable possibilities are confirmed, the system backtracks to the first previous choice point to attempt an alternative path. If there is no such point, a failure is reported.

From a linguist's point of view, this procedure is extremely interesting because it allows the phonology to be represented in exactly the form commonly assumed in grammatical systems: as a system of rules that convert underlying forms into surface forms. If some such model of linguistic performance were confirmed, the relation of linguistic knowledge of the sort that seems theoretically motivated to its implementation would be quite direct. Furthermore, since the system operates in a completely determinate fashion in a top to bottom direction, there is no combinatorial problem deriving from a multiplicative interaction of alternatives as in Kay's proposal. Hankamer's system also operates very efficiently, and so it would appear to present a satisfying resolution of the parsing problem in morphology.

Unfortunately for the generality of this conclusion, the efficient operation of Hankamer's parser is due in essential ways to some accidental properties of Turkish

[5]See Hankamer (1986) for description and discussion.

that do not allow its direct extension to a full range of other languages. First of all, since Turkish is an exclusively suffixing language, the lexical root can always be found reliably at the left edge of the word. Obviously the same procedure could be carried out from right to left instead of left to right for a language that was exclusively prefixing (if there are any such languages), but for languages with non-trivial amounts of both prefixing and suffixing, the problem of finding a starting point for the analysis becomes correspondingly complicated.

Another somewhat adventitious property of Turkish is the fact that in this language, nearly all of the phonological effects that the parser needs to take into account operate from left to right. As a result, the phonological admissibility of the material under analysis up to any given point can be checked exhaustively without requiring any information about material not yet analyzed except its surface phonetic form. Since most phonologies of any real complexity have anticipatory (right to left) effects as well as perseverative (left to right) ones, however, the extension of this mechanism to a representative sample of the languages of the world would require some sort of procedure to keep track of potential effects that can be accepted if and only if their facilitating context arises in material that is analyzed later. This, in turn, contains the seeds of a combinatorial problem when anticipatory effects are substantial.

Further, the phonological effect any Turkish affix has on the material following it is always predictable from its surface form (i.e., neither underlying nor intermediate representations of material already analyzed need to be referred to). This effect is also utilized by Hankamer's parsing strategy, although in many languages representations other than surface forms are necessarily referred to to state the scope of phonological generalizations. The modifications necessary to admit the effects of the underlying form of one suffix on another are minimal, but the incorporation of the effects of intermediate levels of representation require essentially the recapitulation of the entire derivation at each step of the parse.

Finally, Turkish (being very nearly the canonical form of an agglutinating language) has a morphology involving exclusively overt, segmentable affixes which can in principle be looked up in a list. Insofar as a language makes use of non-affixal changes (as in English *sit/sat*), some further mechanisms must be employed to recover the information carried by such operations. This is a problem that is in no way particular to Hankamer's approach — as noted above, virtually all existing parsing strategies rely on the notion that once the phonology proper has been undone, the resulting representation will carry morphological information in the form of an exhaustive segmentation into items on a list of the language's morphemes. Nonetheless, the language particular absence of such nonsegmentable morphology in Turkish contributes to the simplicity of Hankamer's parsing strategy.

All of these points, taken together, indicate that the top down strategy pursued by Hankamer does not generalize directly to a full range of languages without significant revision. It should be emphasized that this criticism is largely irrelevant in the context of a system designed for a practical purpose: if one wants a system that will yield analyses of complex Turkish word forms, this technique exploits the properties of the language fully so as to yield a highly efficient procedure for doing this. If the primary interest is in a parsing procedure as a general model for the morphological structure of natural languages, however, there is a good deal of work still to be done, although the system has a number of very interesting features from a linguistic point of view as well as a practical one.

From work by Kaplan and Kay (1981) which has been widely cited though never published, there originated a somewhat different approach to the phonology. They argued that the effects of a phonological rule can always be represented by the behavior of a finite state transducer (a result developed independently and in more detail by Johnson (1972), whose work was not known to Kaplan and Kay). Importantly, Kaplan and Kay observed that a cascade of such transducers, representing a set of ordered rules, can always be represented as a single transducer. In consequence, since finite state transducers are known to be computationally quite tractable, it would seem that the phonology of any natural language has a representation in the form of one such machine.

The problem is that (assuming the validity of the transducer representation of each rule), the size of the resulting merged transducer can grow exponentially as the set of rules grows. Essentially, when two cascaded transducers have a significant amount of interaction, the result of merging them is a machine whose size is (in the worst case) the product of the sizes of the two simple transducers. Again, as in the pure bottom up approach, a combinatorial explosion results (or at least could result) when a language displays significant depth of ordering. As a result of the recognition of this fact, Kaplan and Kay's observation about the relation between rules and transducers was relegated to obscurity.

In later work, however, Koskenniemi (1983b) observed that in fact many rules are independent of one another within a given grammar, and when this is the case it is possible to model them with transducers that operate in parallel rather than in a cascade. The resulting system has a size equal to the sum, rather than the product, of the sizes of the individual transducers. Koskenniemi proposed that we take this as the general case, and that we deal with violations of it in other parts of the grammar (basically, by representing some parts of some alternations directly in the lexicon rather than as rule governed). This system was further developed by Karttunen (1983) and later work, and has resulted in a very efficiently implemented parser for the morphology of Finnish.

The procedure followed by the KIMMO parser is centered on a representation of the phonology which compares a candidate pair of underlying and surface representations to see if their relation is of the sort sanctioned by the language. This they do by stepping a pair of read-heads simultaneously along the two strings, seeing whether the individual segment-to-segment correspondences are consistent with the set of finite-state transducers (operating in parallel).

Naturally, a fundamental question concerns the way in which the pair of strings to be compared is constructed. The system is intended to be essentially neutral between the task of recognizing a form and that of generating it. When the system is analyzing a form, this is done by building up a candidate underlying form that will satisfy the phonology: start at the left edge, look for a stem, try the morphemes that are in the continuation class of this stem, then add those that are in the continuation class of the last morpheme added, etc., until you have a form whose relation to the given word is well-formed and which exhausts the surface word. The resulting string of morphemes is the analysis. On the other hand, when the system is generating a surface form from an underlying representation, it takes the given underlying string (the analysis) as given and tries (making use of a list of surface segments that could correspond to any given underlying segment) to construct a surface string that will meet the requirements of the phonology. While analysis and generation employ somewhat different strategies, the role of the phonology is essentially bidirectional: it just checks proposed correspondences, rather than creating one form from the other.

The appeal of this (the "lure of the finite state") is that Finite State Transducers are rather restricted in mathematical power, and should be computationally very efficient. The observed behavior of the KIMMO system is certainly consistent with this. However, Barton has shown (cf. Barton (1986), Barton et al. (1987)) that grammars of the class conforming to the requirements of KIMMO-type systems are not guaranteed efficient parsability. In particular, if the grammar contains either unbounded right-context effects or nulls, the parsing problem for such a system is NP-complete (i.e., potentially very hard). Right-context effects and nulls are clearly necessities for linguistically 'clean' accounts of the phonologies of some languages, even though left-context effects are more common than (non-local) right-context effects. Of course, this result doesn't mean that any particular grammar is going to become intractable for a KIMMO system, and if our goals are practical ones, an efficient system of this kind may be just what we want. But if we want to pursue the problem as a form of (general) morphological theory, Barton's result is potentially disquieting.

Aside from the issue of computational tractability, the KIMMO design raises another issue. Since the transducers all run in parallel, comparing underlying and

surface form, intermediate levels of representation are in principle unavailable. Every phonological rule has to express a transparent, surface-true well-formedness condition on the relation between underlying and surface forms. This is the kind of condition that is associated with a phonological theory like that of 'Natural Generative Phonology' (cf. Hooper (1976)), and a substantial majority of phonologists would probably agree that the discussion of that position in the 1970's showed it to be seriously inadequate as an account of phonological structure. Without detailing here the objections that were raised to Natural Generative Phonology, insofar as one takes those objections seriously, they have to count as objections in principle against the KIMMO approach to morphological parsing too.

Again, for practical purposes this may not be significant: alternations that don't conform to the limitations imposed by this approach to phonology can just be listed in the lexicon as if they were suppletive. This is in fact rather extensive in the KIMMO lexicon for Finnish, for example. Although this amounts to an implicit claim that much of the well studied phonology of the language is not in fact systematic (since it cannot be represented as a transparent, surface-true generalization and thus not as one of the parallel transducers of the system), this has no observable consequence for the extent to which the system can produce the correct forms. It is only if this approach is rejected as *theoretically* unsatisfying that it presents a problem.

On the other hand, the KIMMO system makes both the surface and the underlying form available at all points, allowing a kind of global reference ("[a] can correspond to /b/ if it precedes a [c] that's underlyingly a /d/") which is excluded by the purely markovian character of standard sequentially applied rules. The possible linguistic consequences and motivations of this step have not yet been explored.

In sum, the KIMMO approach to morphological analysis has much to recommend it (at least for those languages for which it has been tried — as with Hankamer's approach, this is largely limited to languages that are exclusively suffixing) from a computational point of view. It appears to rest essentially on assumptions that are either discredited or unclear in import from the point of view of linguistic theory, however. If we had reason to believe that human morphological analysis utilizes a procedure that is substantially similar to that of the KIMMO system, this would suggest that much linguistic theory is misguided. To the extent that we accept the results of that theory as giving an appropriate characterization of linguistic knowledge, on the other hand, we have reason to look for an alternative to the KIMMO procedure.

Our result, to this point, is largely negative. The best ways to build a theoretically satisfying account of linguistic structure into a parsing procedure will apparently result (in the general case) in computational intractability; while the only ways to ensure a manageable computation involve restricting in unacceptable ways

our claims about how much of a speaker's knowledge of a language is systematic. Before we go on to suggest a path toward the resolution of this conflict, we discuss some further issues concerning morphological parsing that have not been treated in the existing literature.

1.2 Some General Problems

All of the approaches to morphological parsing surveyed in the preceding section assume that the problem of undoing the phonology and that of analyzing the morphology can be strictly separated. In fact, all of them devote more attention to the problem of undoing the phonology than to the structure of the morphological system itself, assuming that this can be represented as something like a finite state machine describing the admissible concatenations of underlying forms from a lexicon of morphemes. However, there are reasons to believe that this assumption is inadequate. In particular, there is evidence that instead of composing a form morphologically and then submitting it as a whole to the phonology, it is necessary to develop the morphology and the phonology of complex forms in some languages in a more complex, interleaved fashion. Anderson (1975) and others cite cases, for instance, in which some morphological operation crucially must come between two purely phonological operations. A similar conclusion is entailed in a more systematic and principled way by the proposal in Lexical Phonology (cf. Kiparsky (1982), Kaisse and Shaw (1985)) that the morphology and the phonology are interrelated in a cyclic fashion. If some such picture is correct, it follows of necessity that the undoing of the two in a parsing procedure must also proceed hand in hand, rather than being separated as on the models we have considered thus far.

A simple example of this kind can be provided from Javanese (cf. Dudas (1974)). This language has a morphological process that forms the 'elative' (a sort of superlative) of adjectives by raising the last vowel of the stem to a [+High] vowel of the same rounding. This rule interacts with another rule of the language, by which underlying /a/ in final open syllables is rounded to [o]. What is important is that the product of the Elative formation rule when applied to stems whose last vowel is /a/ depends on the applicability of the /a/ → [o] rule:

(3)	/kəmba/	/gampaŋ/	lexical form
	/kəmbo/	—	final /a/ → [o]
	[kəmbu]	[gampiŋ]	Elative formation

Note that in order to obtain the right result, it is necessary to apply the (phonological) rule rounding final /a/ **before** the rule forming the Elative, since /a/'s that

have been rounded raise to [u] in the Elative while those that have not raise to [i]. Now both of these rules can be 'undone' individually — but crucially, the morphological formation must be undone before the phonological rule (starting from the surface form). Thus, the strategy of recovering a phonological underlying form in full before attacking the morphological analysis cannot deal adequately with this case. Examples of genuinely cyclic application from the literature on Lexical Phonology pose much more complicated problems of the same basic sort for this position.

The conclusion to be drawn from this observation is that the analysis of morphology and phonology must proceed hand in hand in an adequate parsing system that hopes to extract all of the linguistically governed regularities of word structure. This is not unnatural in a system that operates in a basically analysis by synthesis mode (such as Hankamer's parser, appropriately modified), but it is hard to see how to incorporate such an approach in a true bottom-up design.

The approaches considered thus far also assume that the central notion of morphology, the claim that two (or more) words are 'morphologically related', can always be expressed as a matter of their 'sharing some morpheme(s)', modulo the effects of the phonology. But some relations are non-affixal (i.e., there won't be a segmentable morpheme for the forms to share), such as the ablaut relation in English strong verbs (*sing, sang, sung*). Furthermore, although the phenomenon has not been much remarked in the theoretical literature, some sorts of morphological relationship have a somewhat non-deterministic character, of the type "forms A and B may be related to one another if they differ only in such and such a way". Importantly, such statements may not involve obligatory phonological rules of any sort, but only quasi-systematic correspondences. This state of affairs is extensively documented for Brazilian Portugese by Lopez (1979); in this language, it comes about from massive borrowing by Portugese from its own history, resulting in the coexistence in the same language of related words representing different diachronic developments. Without attempting to document this situation within the space constraints of this paper, it is clear that both it and the much better known case of non-affixal morphology motivate a more general notion of what it means for words to be morphologically related than is yielded by the usual exhaustive partitioning of words into morphemes.

Another issue in the design of a morphological parser which has gone without discussion is the question of just what kind of result such a parser should return as an 'analysis'. Of course, if words are exhaustively divisible into morphemes, it might seem obvious that an analysis ought to consist of an indication of just what morphemes are present in a given word. That is the account implied by a parser that analyzes a word by indicating where one morpheme stops and the next one begins, and what meaning is associated with each (as virtually all current parsing

systems do).

In the context of an analysis that serves the purpose of a syntactic parser, at least, it is possible to argue that this sort of analysis is both too strong and too weak. It is too strong in the sense that some information (such as linear order of morphemes and the occasional occurrence of multiple markers for the same category) will be present in it which we might want to claim is systematically irrelevant to the syntax. In other words, the syntax cares about what categories are actually indicated by a given word (such as case, number, agreement, etc.), but it does not need (or want) information about just how the word indicates them. On the other hand, the pure morpheme by morpheme analysis usually assumed is also too weak, in that some information may be systematically conveyed not by material that is present in a word, but rather by what is **absent** from it. Instead of a morpheme by morpheme gloss, then, a parser ought really to return:

(4) a. An indication of lexical identity (with a pointer to the syntactic and semantic properties of this lexical item); and

b. A morphosyntactic representation that shows the inflectional categories conveyed by this word-form.

This representation may actually involve some local computation of what properties are present on the basis of the formal markers that can be identified.

These and other issues in the theory of morphology are discussed elsewhere (especially in Anderson (1987), where a survey of the foundational issues in the field is attempted). The point is to notice that a theoretically well motivated account of word structure, in the context of an overall view of the relation of morphology to the other parts of a grammar, involves a number of considerations that have not played much of a role in the morphological parsing literature.

1.3 A Parallel Top-Down Approach to Morphological Parsing

On the basis of the considerations in the preceding section, we might undertake to modify some one of the existing strategies for parsing morphological structure so as to arrive at an account that satisfies the requirements of current linguistic theory in this domain. But even before attempting to do this, we are confronted by the difficulties raised in section (1.1). Simplifying somewhat, analyses such as those of Hankamer or Kay appear to be founded on a linguistically appropriate formulation of the 'knowledge' on which they are based, but to be intractable in computational

terms; while the assumptions that yield computationally manageable descriptions[6] in the KIMMO system exclude much that linguists would argue is systematic from the domain of the principled (as opposed to the listed) in such a system. This could well be taken as a rather severe indictment of the claims of linguistic theory as to what kind of structure ought to be attributed to speakers' knowledge of the phonology and morphology of their languages. It might of course simply be the case that descriptions of linguistic competence and of actual processing do not relate directly to one another; but at least *prima facie*, it ought to be unsettling to conclude that knowledge organized in the way linguistic theory appears to indicate seems not to be appropriate as the basis of a computational procedure to analyze the structure of linguistic objects.

If what seems to be the linguistically motivated description is to be incorporated fairly directly into a parser, as for instance in Hankamer's approach, what we might want the procedure to do (at least schematically) is first to find in the lexicon the root (or derived stem) on which the form is built; and then to explore the paradigmatic space of that lexical item to find the word under consideration. Let us assume (following Anderson (1987) and work cited there) that at least insofar as inflectional morphology is concerned, the dimensions of this space are given by the structure of Morphosyntactic Representations in the language in question. Each such MSR is an indication, in the form of a complex symbol,[7] of the externally relevant properties of the word indicated by its form. The parser must be given a (language particular) definition of the well-formed MSR's associated with words of a given class, together with the system of morphological and phonological rules of the language that yield surface forms from their representation as a pair $\{\mathcal{S}, \mathcal{M}\}$ consisting of a lexical stem $\mathcal{S}$ and a MSR $\mathcal{M}$. On the basis of that information, the exploration of the paradigmatic space of an identified lexical item could consist of trying out various possible MSR's to see which one(s) might yield the word under consideration from the (previously identified) lexical stem.

On this account of how the problem is to be decomposed, we can note that the total complexity of the lexicon of a morphologically elaborate language can be factored into two components, corresponding to these two parts of the the problem: the number of stems (or 'basic' lexical items), multiplied by the number of forms in the paradigmatic space of each. The total set of possible surface words in a language, that is, may be very large. To analyze a particular word using an unstructured analysis by synthesis procedure, which would implement the linguisti-

[6]Recall, however, the demonstration in Barton et al. (1987) that even KIMMO systems do not guarantee efficient parsability to the extent that was originally believed.

[7]As discussed in Anderson (1987), these are largely but not entirely unstructured collections of features. See that work and references cited there for further elaboration of the notion of MSR.

cally motivated rules of the language in a more or less direct way, it is necessary to try out vast numbers of alternatives that are not ultimately successful. Whenever this happens, it is necessary for the parser to backtrack and start over again. As a result, the time necessary to perform such an analysis grows with the the complexity of the linguistic system, and not simply with that of the word being analyzed. Now this seems to be quite unrealistic, as a claim about psychological processing: Speakers of a language such as Finnish, whose total set of words is much larger as a function of the large paradigmatic space occupied by each lexical item, do not seem to take correspondingly longer to recognize individual words than do speakers of morphologically much simpler languages, like English.

How might we eliminate the apparent consequence of an analysis by synthesis, namely that the large number of incorrect alternatives that must nonetheless be explored leads to massive backtracking and a concomitant explosion of the time required for recognition? One obvious path to such a result would be to explore the set of potential analyses at any given point in the procedure not one after another, with backtracking, but simultaneously and in parallel. Of course, if the computational resources required for such a step are unacceptably massive, we have simply exchanged an unintuitive result about processing time for an equally unfortunate result about processing space. Perhaps, however, it is possible to decompose the problem in such a way that a more or less manageable number of alternatives is under consideration at any given time; and that as some of these prove untenable, the processing capacity previously devoted to them can be re-allocated to more extensive consideration of the remaining viable hypotheses.

We propose that just such a decomposition is implicit in our formulation of the problem above: first we wish to identify the lexical root, and then once we have found the root(s) that might underlie the word, we want to probe the language's paradigmatic space around each of them. Suppose that the number of lexical bases in a language is of the order of 64,000 (roughly the size of the concise edition of *The Random House Dictionary*). That number might be taken as the size of the parser's initial hypothesis space, concerning the possible root; but of course a very quick check would show that the vast majority of those bases could not possibly underlie any given word being analyzed. Once a candidate set of bases has been selected, the dimensions of the language's paradigms (assumed to be complex) can be considered; but the number of possible values along any of those dimensions is likely to be fairly small, so that the number of new alternatives generated by considering the range of values for a given dimension will still be small compared to the number of previous hypotheses that can be rejected.

It might be objected that of course a parallel procedure will economize on time, because it involves many more computations during a fixed amount of time than

a serial process. In this vein, it has been shown in the literature on algorithms for search that parallel (as opposed to serial) processing results only in a linear improvement in speed. If the search space is potentially unbounded in significant ways, the shift to parallel processing does not then change the qualitative dimensions of the problem. Barton et al. (1987) similarly argue that parallel processing does not change in any interesting way the range of languages that are efficiently parsable by a given approach, because a linear speedup factor does not cope effectively with exponential growth in the search space.

This is not the point, however. The procedure just suggested can be expected to achieve computational efficiency not simply by multiplying the resources devoted to the problem, but by exploiting the fact that the systematic structure of a natural language's word stock allows us to explore it in a way that keeps the growth of alternatives tractable in purely local terms. Having found a possible lexical base, most of the computational resources originally dedicated to distinct lexical possibilities can be immediately re-allocated to exploring the set of words that might be founded on that base. Since the procedure involves no backtracking, many possible paths can be pursued in a non-exclusive fashion. One that fails to be confirmed may have a temporary cost in processing resources, but none in added time.

The effectiveness of this response to the objection that parallelism yields only a linear time advantage rests on the fact that natural language morphologies are not really unbounded in the relevant ways. Only if the number of alternatives being pursued at any given point (initially, the size of the lexicon, and later, the number of alternative morphological developments from the remaining viable bases) becomes large relative to the number of processors does the effect of parallelism reduce to a simple linear advantage in speed. A combination of the fact that there are only limited numbers of possibilities at any given point with an efficient pruning procedure can keep the growth of alternatives within reasonable limits. We claim that in actual natural languages, the order of parallelism that is required to achieve a qualitative advantage, while large, is by no means unbounded, and in practice well within the capacity of currently available parallel computers.

On this line, we can identify two problems that the parsing strategy must solve if it is to recognize words on the basis of a linguistically satisfying description of the lexicon, morphology, and phonology of the language. First it must find the lexical base of the word. This procedure is, as we noted above, comparatively trivial in a language like Turkish, whose exclusively suffixing nature ensures that the left edge of the root will coincide with the left edge of the word. If a language has non-trivial amounts both of prefixation and of suffixation, however, such a result will not be guaranteed, and we need some richer procedure to ensure that the analysis can get off the ground without extensive backtracking. Secondly, we must provide a way in

which the paradigmatic spaces of candidate lexical roots can be examined without backtracking that expands exponentially as a function of the number of dimensions of this space.

An appropriate massively parallel computational architecture within which to develop such a parser appears to be provided by the 'Connection Machine' (see Hillis (1985) for a description of this machine and its properties), and we hope to explore the problem with this hardware. A prerequisite for efficient use of a 'Single Instruction, Multiple Data' (or SIMD) device such as the Connection Machine is that a problem be factored into a single series of instructions that can be executed in parallel over a large space of data elements, but the morphological parsing problem can be given this structure without apparent loss of generality. When the problem is viewed in this way, the data objects are the individual candidate hypotheses about the word's analysis that are viable at any given point. Each such hypothesis has two aspects:

(5) a. A proposed (perhaps only partial) representation as an abstract $\{\mathcal{S}, \mathcal{M}\}$ pair as described above; and

b. A (perhaps partial) representation of the surface form that would correspond to this pair.

There are two sorts of operation that can be performed across the entire set of such hypotheses:

(6) a. Candidate $\{\mathcal{S}, \mathcal{M}\}$ pairs can be sent through the morphological and phonological rules of the language to derive their surface form (to the extent possible at a given point in the development of hypotheses); and

b. Candidate surface forms can be compared with the actual word under analysis to see if they are grossly compatible with it.

The comparison referred to in (6b) above can be performed quickly and efficiently by a set of finite state transducers operating in parallel, specifying in a language-particular fashion the range of deviations between underlying and surface forms. This specification is intended only as a rough pruning procedure for hypotheses, and not as a systematic confirmation of the correctness of any one; it can therefore operate in a 'permissive' fashion, without checking for the validation of contextual effects which could lead to backtracking.

The strategy to be followed in pursuing the analysis of a given word is as follows. The Connection Machine consists of 64,000 interconnected processors.[8] When an

[8] Actually, on the hardware with which it is intended to pursue this research, only 16,000 physical

analysis is initiated, the control program assigns one processor to each lexical base, allowing that processor simply to assume that its base is the correct one for the form in question. The correctness of this assumption is then verified locally by applying the comparison procedure sketched above.

Insofar as a given hypothesis (at any stage of the procedure, including this initial lexical search step) is shown not to be viable, the processing resources allocated to it can be freed for other uses. This corresponds to the backtracking step in a serial search procedure, but simply means in this context that some of the parallel searches are to be abandoned in favor of others. For those hypotheses still in contention, processors are allocated to explore individually the various possibilities within the (remaining portions of each one's) paradigm. Trying a hypothesis means applying those morphological rules (together with their phonological consequences) that would follow from it, and checking (again, permissively) to see if the result might be consistent with the form being analyzed. When all of the paradigmatic dimensions have been exhaustively searched for all of the roots that remain viable, the remaining hypotheses can be run through the rules to see which (if any) match the form in question *exactly*. The resulting $\{\mathcal{S}, \mathcal{M}\}$ pair(s) can be considered the analysis (or analyses) returned by the parser.

Obviously, many of the details of this procedure remain to be developed at this point, though it appears that at least for the range of structures that occur in inflectional morphology, the dimensions of the problem do not exceed the resources available. If this is indeed the case, a number of important conclusions would follow from that fact. For one, such a parser should be able to operate in something like linear time as a function of word length. This would be in accord with the psychological literature on word-recognition behavior as far that literature goes. Given that at least a gross match might be obtained between the behavior of the parser and that of actual human subjects, we could begin to redeem the promises made at the beginning of this paper to confront an explicit model of human linguistic performance (in at least this one area) with the observable facts of natural language processing.

Another area in which this parsing strategy seems likely to yield results that can be compared with observed data in the psychological literature is in the area of lexical priming. It has been noted in a great many experiments that when a word is recognized, a number of other words related in various ways to it are simultaneously 'activated' to a greater or a lesser degree. A parallel search strategy such

processors will be available. Each of these can transparently simulate a larger number of virtual processors, however, at a cost in the amount of memory available to the individual processes. For the problem considered here, this fact is irrelevant.

as that outlined above necessarily replicates this property, since a number of possible hypotheses are always being entertained simultaneously at any given time. By examining the class of words that appear to be related in terms of lexical priming experiments, we could hope to refine the model's picture of the course by which the hypothesis space is structured and progressively narrowed.

Furthermore, the parser in question could be coupled with a syntactic analysis procedure proceeding in parallel with it, in such a way that the two modules of the overall analysis could dynamically take into account information exchanged with one another. Thus, the syntactic analysis of one part of a structure constrains the class of viable morphological hypotheses at some other point to those consistent with (at least one of) the evolving syntactic possibilities. For instance, when the properties of the subject NP are identified with sufficient precision, these limit the agreement properties of the main verb of the clause. Since each of the components of the analysis process — the syntax and the morphology — are engaged in a process of continued hypothesis formation and refinement, they should be able to make use of one another in speeding the process of eliminating impossible or inconsistent analyses.

Finally, and perhaps most importantly, the representation of the grammar in the parser can have all of the characteristics that appear to be motivated on theoretical grounds (cf. Anderson (1987)) — in particular, it can be based on a representation of morphological structure as a set of word formation rules rather than as a dictionary of morphemes. The rules of the morphology and of the phonology can encompass regularities of some abstractness (as required by linguistic theory, but precluded by parsing strategies tied too closely to direct correspondences between underlying and surface form, such as KIMMO). If such a parser can be developed as planned above, it would thus constitute an explicit model of linguistic performance (at least with respect to word recognition) which incorporates in a rather direct form an account of the structure of linguistic knowledge that corresponds to the linguist's notion of competence.

1.4 Conclusion

In earlier sections, we surveyed a number of approaches that have been taken in the literature to the problem of parsing morphological structure, and noted some areas in which they appear to be deficient. The most important difficulty posed by these deficiencies, however, is not merely the computational one of efficiently recovering morphologically encoded information for practical purposes; rather it is the apparent conclusion that for reasons of principle, an efficient parser could not

incorporate in any direct way knowledge about linguistic structure with the form usually ascribed to that knowledge by theoretical linguists. If this is indeed true, it would indicate that linguistic theories might well be systematically misguided in fundamental ways.

By presenting an alternative approach to the parsing problem which shows promise of resolving this apparent contradiction between the nature of competence and strategies of performance, we thus potentially validate the significance of the class of linguistic descriptions entertained by most theoreticians of language. In terms of such a model, the details both of the theoretical description and of observed human processing behavior are directly relevant: the former because they determine the actual set of rules that the model makes use of in constructing and testing hypotheses; and the latter, because the behavior of the system can be compared in rather fine detail with that of human subjects. The extent to which such a procedure is potentially a genuine and principled model of linguistic performance is therefore quite striking.

The essence of this model is of course its reliance on parallel processing to avoid the problem of backtracking. Here we are surely on fairly certain ground. Little of significance is known about the properties of the actual neural 'hardware' with which human speakers process language, or at least little that would enable us to constrain the class of procedures that might be applied to this task. One thing that is known, however, is that human cognitive processing is unlikely to make use of long and complicated procedures applied very quickly one step at a time (comparable to the operation of standard serial computations). Rather, human neural computation is apparently quite slow (at least in comparison with the behavior that can be extracted from silicon), but involves massive parallelism, with lots of things going on at the same time. In replacing a serial computation with a massively parallel one, the parsing strategy outlined above is presumably moving at least grossly in the direction of greater conformity with the sorts of processing that might be going on in the nervous system.

This observation, in turn, suggests another reason for pursuing such an analysis. Linguistics is by now a highly formal field, with rather explicit computations corresponding to most of its basic constructs. Central areas of the field, such as formal language theory, the notion of derivations relating underlying to surface form, etc., are routinely presented in the form of explicit algorithms when they are to be made precise. Our very concept of what constitutes a possible (or a plausible) rule, grammar, constraint, etc. is usually formed on the basis of its translation into some explicit procedure for effectively computing the nature of some class of linguistic objects. Now in fact the computations which are usually envisioned are exclusively serial ones: they proceed one step at a time, and it is the specification of the nature

of these steps and their interrelations that constitutes much of linguistic theory.

Suppose, however, that the processes underlying human linguistic performance are actually massively parallel, relatively slow, and locally rather simple, rather than serial and made up of an intricate sequence of many steps executed relatively quickly. In that case, our underlying sequential metaphors might be radically unsuited to the capturing of linguistic reality; and we would be well advised to seek a way of recasting the foundations of the field in terms of a more appropriate set of computations. In the present instance, the result may well be a validation, rather than a radical revision of accepted results. Once the kind of knowledge phonologists and morphologists attribute to speakers is interpreted in terms of parallel rather than sequential processing, it may be that this is not as remotely related to linguistic performance as is sometimes thought. Nonetheless, it may well be that there are important novelties and surprises in store for us when we attempt to take seriously the notion of parallel processing. It is most likely to be through computational linguistic research, now that genuinely parallel hardware such as the Connection Machine is coming to be available, that our horizons stand to be expanded in this possibly fundamental way.

Chapter 2

Parsing Morphology Using Definite Clauses in Prolog

Sean Boisen

2.1 Introduction

This paper introduces the DCM model, an extension of Definite Clause Grammars (Pereira and Warren (1980), henceforth DCG's) for parsing morphology. [1] The basis of the DCM formalism in logic programming provides greater clarity and power than previous approaches to morphology (primarily the "two-level" models, Koskenniemi (1983b), Karttunen (1983)), while preserving many of the advantages of these systems. While relatively efficient to operate, the two-level model incorporates a number of features which are questionable from a linguistic standpoint. The morphosyntactic component is equivalent to a finite-state grammar, using continuation classes for each lexical element. This treatment requires substantial duplication for dependencies between non-adjacent formatives: these can be handled more suitably with a context-free system (this point is discussed in more detail in Karttunen (1983)). The relationship between surface and underlying phonological forms is direct (hence the "two-level" label): thus there are no intermediate forms,

[1] This work has been supported in part by grant number BNS-84-18277 from the National Science Foundation to the University of California, Los Angeles. Many of the issues presented here have been discussed at length with other colleagues. Foremost among these are Steve Anderson and the participants in his 1986 seminar on computational morphology at UCLA, especially Tom Cornell and Karen Wallace. Naturally I alone bear the responsibility for errors in my views.

and no interactions between phonological processes. The phonological representations are based on unitary and non-hierarchical segments: there is no provision in current models for phonological features or non-linear representations (as, for example, in the theory of CV phonology presented by Clements and Keyser (1983)). Phonological process are encoded as finite-state automata, which do not directly correspond to the generalizations they depict.

A more substantive critique of the two-level model from a linguistic perspective is outside the scope of the present article. Nevertheless, it is clear that there are several respects in which a more suitable model of morphological processing can be devised. The DCM formalism is directed toward this end.

2.2 Parsing Morphology

Since the DCM model is within the spirit of DCG approaches to syntactic parsing, it inherits most of the advantages of that technique. DCG's have proved useful for a wide variety of parsing applications in the syntactic domain, for reasons argued convincingly in Pereira and Warren (1980). The fundamental technique is the construction of difference lists of words: the fundamental advantage is that such programs are directly executable as Prolog programs, using a suitable translator which is easily constructed. The remainder of this article assumes familiarity with the DCG formalism.

Since the requirements of morphological processing differ somewhat from those of syntax, we review here some of the capabilities that a logic-grammar approach to morphology must provide. We assume throughout the Extended Word-and-Paradigm (EWP) model of morphology (Matthews (1972), Thomas-Flinders (1981), Thomas-Flinders (1983)). As a crucial feature of this model, we expect a morphological parse to return only those features which are relevant to the syntax. This includes most inflectional characteristics (e. g. number, tense, case), but not derivational composition (for example, the fact that a verbal form is derived from a noun). Similarly, we do not return or even expect to find a strictly concatenative partitioning of the form into component parts: within the EWP model, this is not always possible, and clearly it is irrelevant to the syntax.

Morphological parsing is further distinguished from syntactic processing by the requirement that forms satisfy not only lexical and morpho-syntactic constraints, but phonological ones as well. The primary concern of the present research is modelling linguistic theory: for this reason, we process actual segments, rather than the orthographic characters of most two-level descriptions. While there are certainly similarities between these two approaches, there is little of linguistic interest in han-

dling the essentially arbitrary spelling conventions of English, for example (as in Karttunen (1983)).

Our system is further distinguished from two-level model in its treatment of phonological rules. Since a principal tenet of the two-level approach is that phonological processes can be treated as "shallow", phonological changes in these models are "undone" throughout the parse, directly relating the surface string to an underlying one. The DCM model, however, has a considerably more complex phonological component, including at present phonological features and a CV representation. Furthermore, as opposed to the two-level approach of rules encoded as finite-state automata, DCM rules are applied directly to forms, allowing cyclic application and rule interactions of the kind standard in generative phonology.

This kind of phonological processing is computationally much more "expensive" than operating a finite-state machine. Additionally, the addition of more complex representations and the possibility of rule interaction increases the search space dramatically, producing a combinatorial problem for any approach which works from a surface form back to its underlying representation. We therefore use a heuristic approach which hypothesizes a set of underlying candidate forms: each of these is then processed fully, from an underlying representation to a surface realization, which is then tested against the input string. This constitutes a working hypothesis, subject to empirical confirmation, that it is more efficient to "guess" a limited set of possibilities and test them fully, than to try to undo a complex set of phonological restrictions (the two-level approach).

Clearly the success or failure of this kind of approach depends crucially on the power of the heuristic device, which we have called "fuzzy phonology". In the present system, the hypothesizing mechanism is a streamlined form of the phonological rules, encoded for efficiency as a standard two-level automata, which admits permissible structural changes without, in general, testing structural descriptions (some complications have been omitted from this discussion, of course). The "fuzzy phonology" therefore overgenerates, but is guaranteed to include all actual underlying forms. As long as the set of candidates generated is reasonably small, this should prove to be a more tractable approach to incorporating a robust phonological component.

In the remainder of this article, we will therefore assume that we are working with the output of the "fuzzy phonology": given a candidate underlying form, then, we must ensure that it satisfies morphosyntactic constraints and is a valid lexical item. We must also use the phonological rules to derive a surface string, which is then tested against the input string. If the strings match, the morphosyntactic features are returned: otherwise, the parse fails.

2.3 The DCM formalism

As mentioned above, the DCM formalism is an extension of the DCG approach. One feature is the separation of the lexicon from the grammar itself. This is desirable, if only for purposes of modularity: it is also important in the DCM system because of the need to access phonological representation. Rules at the "raw" (untranslated) level must be able to refer to what are normally the "hidden" arguments, for purposes of satisfying various co-occurrence restrictions. A simplified example (taken from the author's work on Khalkha Mongolian) is presented in figure 2.1.

```
noun([-, singular]) -->
       lex(n_stem, Stem),
       {n_final(Stem)},
       lex(plural, n_type, Plural).
noun([-, singular], Stem, S) :-
       lex(n_stem, Stem, Plural),
       n_final(Stem),
       lex(plural, n_type, Plural, S).
```

Figure 2.1: A sample DCM rule, illustrating the need for accessing hidden arguments, together with the Prolog code produced for this rule by the DCM translator.

In Mongolian, there are several plural markers, the choice of which depends on phonological characteristics of the stem. The bracketed predicate `n_final` tests the stem to see if it meets the phonological requirements of this particular plural marker (here lexically marked as `n_type` for expository purposes). The translated form indicates that the lexicon is based on difference list notation: an entry for a noun stem might be

```
lex(n_stem, [d,e,l,x,i,i|Tail], Tail).
```

This rule cannot be written in a standard DCG formalism (except, of course, by writing all the object code inside brackets, a pointless exercise). If there were no phonological restrictions, a typical DCG rule might be like the one presented in figure 2.2: this sort of rule still fails to separate the lexicon from the grammar, however. [2] This notation also provides no means for accessing the actual form of

[2]The standard trick for building a dictionary into a DCG is a rule like

```
n --> {lex(noun, N)}, [N].
```

This won't work if the lexical item is not atomic, however, since the standard DCG translator cannot process terminal arguments which are atomic. That is, while the interpretation of rules like

```
noun([-, singular]) -->
       n_stem,
       plural(n_type).
n_stem --> [d,e,l,x,i,i].
plural(n_type) --> [+,d].
```

Figure 2.2: A standard DCG-type rule for the phenomenon above (without the phonological test).

the noun stem, since the argument representing it is only inserted in the course of the conversion to actual Prolog code. Since this is frequently important in morphological systems, the DCM extension becomes significant. Note that the resulting translation is still a typical difference list program: a noun which is `[-, singular]` is composed of a lexical element which is a noun stem, followed by a plural of the `n_type`, provided that the stem is n-final. In the DCM system, then, `lex` is a reserved word, which triggers the unique translation indicated in figure 2.1.

Other rules are translated like ordinary DCG's. This allows straightforward inclusion of morphological grammars in DCG-based syntactic systems, by interfacing the morphological parse to the syntactic one. This interface primarily concerns phonological processes. First, the terminal elements of the syntactic module must be converted from atoms to lists, providing segments for the morphological system to operate on. A small fragment of a grammar to illustrate this is presented in its translated version, together with the DCM notation, in figure 2.3. Under

```
% the "raw" version
n(Number, [Word|S], S) :-
     explode(Word, Segmented_word),
     noun(Number, Segmented_word, []).
% a grammar rule version
n(Number) --> word, noun(Number).
```

Figure 2.3: The DCM relationship between the syntactic atom n and the morphological string noun.

```
n --> {lex(noun, N)}, N.
```

is clear (under the assumption that N is a list), such rules are not well-formed in the DCG notation.

this scheme, "word" becomes a reserved identifier for the DCM translator, which produces the translation in figure 2.3. Note that this separation of the syntactic and morphological domains corresponds to the general unavailability of phonological information in the syntax, since individual characters are only directly accessible within the word domain. The `explode` predicate here maps between an atom and a list of its characters, as in

```
explode(cat, [c,a,t]).
```

There are other characteristics of the phonology which are not dealt with here, for example the fuzzy phonology, and how simple character lists are mapped onto hierarchical CV representations. In addition, the output of the morphological parse as treated here must include a new derived form which can be tested against the input string. This example is therefore a simplification of the actual DCM system: the extensions to the DCG notation should be clear, however.

2.4 Advantages of the DCM formalism

As illustrated above, the DCM system has the important advantage of being able to test phonological information in satisfying a parse. This is significant for languages like Khalkha Mongolian, where there are several alternative forms for the plural marker, most of them selected on the basis of phonological characteristics of the stem (Poppe (1970)). The treatment of this in a two-level model requires arbitrary lexical classification of all noun stems according to the plural marker that they take. This is clearly unjustified when there are obvious linguistic generalizations concerning affix co-occurrence (namely phonological considerations). It also unnecessarily complicates the rest of the nominal morphology: for example, the rules governing the occurrence of case markers do not divide according to the same lexical classifications, but are determined by different phonological properties of the stem.

A related advantage of the DCM formalism is important within the EWP model of morphology. Under this theory, the "exponent" (i. e. marker) of a particular morphological category need not be the additional of some affix: instead, phonological change can reflect a morphological category. Umlaut and ablaut processes are well-known examples of this. There is no way to capture this "process" approach in a two-level model, other than lexicalizing it. The DCM model has access to the phonological form of the stem, however, and can therefore directly represent this kind of process in a natural way.

To briefly illustrate, German marks the plural of certain nouns by umlaut: thus [nɑxt] is "night", and [nɛxtə] is "nights", where the stem vowel has been fronted

(and [+ə] added). A DCM rule to capture this is presented in figure 2.4. Given a form like [nɑxtə] (hypothesized by the fuzzy phonology), this rule returns the umlauted form (for eventual comparison with the input string), as well as the fact that it is plural. The general form of the `umlaut` predicate should be clear from

```
noun([-, singular], New_stem) -->
        lex(n_stem, Stem),
        {umlaut(Stem, Umlauted_stem)},
        [+, @],
        {append(Umlauted_stem, [+, @], New_stem}.
```

Figure 2.4: A DCM representation of German umlaut.

the figure. While there is certainly additional complexity to be accounted for in a robust treatment of German morphology, this short example indicates the nature of such a treatment in the DCM model.

The umlaut example suggests another application of DCM grammars: capturing local morpho-phonological processes. It is often the case that some phonological change accompanies the marking of a morphological category. Furthermore, these changes are essentially restricted to a particular morphological environment, that is, they are conditioned by morphological elements in addition to the usual phonological circumstances. For example, in Khalkha, the final diphthong of certain noun stems reduces to ə when it is followed by one of the plural markers: similar diphthongs which take a different plural do not undergo this reduction, and in fact it only occurs in this one environment. The appropriate linguistic generalization here seems to be that this particular alternation is conditioned by a specific formative: the ordinary phonology should therefore not be burdened with it. Analogous to figure 2.4, we can include a predicate in this one rule to handle the alternation. While we don't want to treat truly general processes (i. e. those whose conditioning environments are strictly phonological) as local exceptions, we also don't want to treat diphthong simplification as a general process when it appears to be limited to this one environment.

2.5 Conclusion

We have outlined above a formalism which, we have argued, is more suitable than current models for handling the kinds of morphological processes which occur in the world's languages. Space constraints here prevented a complete discussion of the phonological component: it is an advantage of the DCM model, however, that the

separation of lexical material from the grammar allows us to abstract away from many details of phonological representation. As with all formalisms, the ultimate justification of the DCM model is the ease with which it allows the user to concentrate on the details of the grammar. We expect to continue using DCM grammars to write substantive descriptions of actual morphological systems in future work.

Chapter 3

Pro-KIMMO: A Prolog Implementation of Two-Level Morphology

Sean Boisen

3.1 The two-level model

The two-level formalism was designed by Kimmo Koskenniemi, as described in Koskenniemi (1983b), a revised version of his dissertation.[1] Originally implemented in Pascal, it incorporates a model of morphological analysis and synthesis which uses finite state transducers to describe Finnish, a language with fairly complex morphology. Later work by Lauri Kartunnen and his students at the University of Texas at Austin extended Koskenniemi's work by developing a LISP implementation of the program (which they named `KIMMO`, after its originator). They also broadened the base of languages to which the model was applied, developing "two-level model" morphological descriptions for English, Japanese, Rumanian and French. Both of these implementations incorporated the ability to perform recognition and generation of morphological elements using the same description, one of the strong points of Koskenniemi's approach. The current implementation, `Pro-KIMMO`, only includes the recognition capacity, although it would not be too difficult to expand the

[1]This work has been supported in part by grant number BNS-84-18277 from the National Science Foundation to the University of California, Los Angeles.

program to generate forms as well. With this one exception, all three implementations appear to embody the same basic functionality and approach to morphological description.[2]

3.1.1 Why Pro-KIMMO was designed

There were several motives for building a Prolog version of the KIMMO model. One was to develop an accessible program for students to gain hands-on experience with morphological processing. Particularly because of our interest in developing successors to the two-level model, access to some implementation of the current "state of the art" was essential. The decision to use Prolog was based on the recent surge of interest in declarative programming, and a desire to evaluate this language's suitability for morphological processing. Since we did not have a good implementation of LISP available at the time the project was started, this was another factor in Prolog's favor. In addition, it was felt that the non-directive aspect of Prolog processing might prove useful in handling the generation/recognition symmetry in KIMMO.

3.1.2 Input and output

As a morphological recognizer, Pro-KIMMO is designed to take complete words as input and produce a listing of the component parts. A sample run of the program is presented in figure 3.1.

```
?- recognize("cats").
Recognized string: cat+s0
categories:        [root, n, c2]
feature(s):        [[], N PL, []]

yes
```

Figure 3.1: Sample input and output from Pro-KIMMO.

The recognizer is invoked within the standard Prolog interpreter by attempting the recognition of a target string as a goal. In addition to the ubiquitous Prolog success indicator of "yes", it displays (as a side effect) the underlying string to which it was matched, the categories of the elements it recognized, and any features

[2]Pro-KIMMO also makes no provision for merging automata: this is an efficiency consideration, not a crucial feature of the model, however.

of these categories. The underlying string is not segmented into its component formatives. Following the example in figure 3.1, the four letter sequence "cats" was matched to the six character string above. In the lexicon, the string "cat" is classified as a root element with no additional features. The lexical category for the string "+s" is [n]: this indicates that it is a nominal suffix.[3] This case should be distinguished from that of a noun itself, whose lexical category is [root]. This string has the additional features of plurality (PL) and nominality (N): in this particular description, roots are only explicitly marked as nouns when a (possibly null) number marker is added to them. The final character "0" indicates that a null string has been matched in the recognition process, in this case the empty counterpart to the possessive marker (c2), which has no additional features. While it may seem peculiar at first to explicitly depict the matching of an empty element, such a representation is preferable because it clearly indicates when null strings are matched. This is a fairly common occurrence in some two-level descriptions of languages with more complex morphology.[4] It should be noted that this is a minor departure from the output format described in Karttunen (1983).

This basic approach is further exemplified in figure 3.2. The first token illustrates handling the exceptional form of the plural suffix, here `-es` rather than simply `-s`.[5] Naturally, the formation using the regular plural ending should fail in this case, as the second example indicates. The third example in figure 3.2 demonstrates the recognizer output for a string which is not overtly marked for number (hence a null is matched for the singular "ending") but which is marked as possessive.

A few other points can be made with respect to the format of input and output under `Pro-KIMMO`. Figures 3.3 and 3.4 show that ambiguous forms return more than one result, utilizing the automatic backtracking feature of Prolog. This is true both for lexical ambiguity (figure 3.3) and for morphological ambiguity (figure 3.4, `undoable` parsed as `undo + able` or as `un + do + able`).

On the nature of the "underlying" strings, the somewhat erratic use of the plus sign underlines the fact that it is not exactly a marker of formative boundaries, contrary to its common use in generative phonology. It is best viewed as a diacritic character, necessitated by the finite state approach used in `KIMMO` and `Pro-KIMMO`. Since none of the orthographic rules in Karttunen and Wittenberg (1983) pertain

[3]It should be observed that the particular choice of categories, features, and labels used here are straightforwardly those developed in Karttunen and Wittenberg (1983), and that they are independent of `Pro-KIMMO` itself.

[4]For example, the author's (unpublished) work on a two-level model of Khalkha morphology.

[5]The specific description set forth in Karttunen and Wittenberg (1983) is based on orthographic conventions, rather than strictly morphological ones. The relevance to this latter domain should be fairly clear, however.

```
?- recognize("foxes").
Recognized string: fox+s0
categories:        [root, n, c2]
feature(s):        [[], N PL, []]

yes

?- recognize("foxs").
no

?- recognize("fox's").
Recognized string: fox0's
categories:        [root, n, c1]
feature(s):        [[], N SG, GEN]

yes
```

Figure 3.2: Possessive and plural forms of fox.

```
?- recognize("are").
Recognized string: are
categories:        [root]
feature(s):        [V PRES SING 2ND]

Recognized string: are
categories:        [root]
feature(s):        [V PRES PL]

Recognized string: are
categories:        [root]
feature(s):        [AUX]

yes
```

Figure 3.3: Multiple parses returned for are illustrating lexical ambiguity.

```
?- recognize("undoable").
Recognized string: undo+able
categories:        [root, ab]
feature(s):        [[], VERB ABL]

Recognized string: undo+able
categories:        [root, root, ab]
feature(s):        [NEG, [], VERB ABL]

yes
```

Figure 3.4: Multiple parses returned for undoable.

```
epenthesis ("cc", [2, 2, 2, 2, 0, 1]).
epenthesis ("hh", [1, 3, 1, 3, 0, 1]).
epenthesis ("ss", [4, 3, 3, 3, 1, 0]).
epenthesis ("SS", [3, 3, 3, 3, 0, 1]).
epenthesis ("yi", [3, 3, 3, 3, 0, 1]).
epenthesis ("+e", [0, 0, 5, 5, 0, 1]).
epenthesis ([43, 0], [1, 1, 6, 6, 0, 1]).
epenthesis ("==", [1, 1, 1, 1, 0, 1]).
finality (epenthesis, [true, true, true, true, false, true]).
```

Figure 3.5: Epenthesis automaton for English, modified slightly from Karttunen and Wittenberg (1983), p. 227.

to this particular environment, there is no particular need to include a formative boundary symbol here: it is used elsewhere primarily to make the transducers work out properly, as discussed below.[6]

3.1.3 Pro-KIMMO as a finite state transducer

Much of the appeal of Koskenniemi's model lies in its use of finite state transducers to describe morphological relationships. While a full consideration of the appropriateness of a finite state system is beyond the scope of this paper, some general comments can be made. First, because a finite state grammar is relatively simple computationally, it is possible to implement a very efficient processor.[7] While much attention is given in Koskenniemi (1983b) to the front end of the system as an alternative formalism for morpho-phonological rules, the ultimate purpose of these rules is the production of a state table like that presented in figure 3.5.[8] This is one of the six automata presented in Karttunen and Wittenberg (1983): the full set is shown in appendix B.

[6] Karttunen and Wittenberg (1983) indicate in their discussion of the un prefix (pp. 221–223) some of the complications that lead to this treatment.

[7] The figures presented in various papers by Koskenniemi indicate that the Finnish system handles a reasonably complex word in about 0.1 CPU seconds (Koskenniemi (1983a), p. 685; Koskenniemi (1984), p. 181).

[8] In fact, we made no use whatsoever of the extensive formalism Koskenniemi describes. In the absence of some automatic means of converting such descriptions into automata, they are more of a nuisance than a help.

The structure of the automata in `Pro-KIMMO` is somewhat different from that used in Koskenniemi's Pascal implementation, as well as in `KIMMO`, due to the different conveniences of Prolog. The **epenthesis** clauses each designate the state transitions (the second argument) for a particular input pair (the first argument). The first character in the input pair is the lexical member and the second is its surface equivalent. The members of the transition list designate the state after reading a pair: 0 here represents failure, sometimes referred to as the "trap". The current state is represented implicitly by the order of list, that is, there is a covert indexing in the list.[9] So, using the first clause as an example, if the current state is 5 and the pair ⟨c.c⟩ is read, the state is set to 0 (indicating failure due to a blocked automata). The automata can be visualized as two-dimensional matrices whose rows are pairs being read by the transducer and whose columns represent the current state in the automaton.[10] The seventh **epenthesis** clause uses an alternative formulation which Prolog provides for representing character strings, namely a list of the decimal ASCII codes. This is used in this example to represent the correspondence between the lexical symbol "+" (ASCII code 43) and a surface null (ASCII code 0, which cannot be represented in a string notation in Prolog). There are a variety of notational conventions used in `KIMMO` to approximate the notion of phonological classes, which will not be reviewed here: the pair ⟨S.S⟩ represents this abbreviatory device. Likewise, the "=" symbol represents a special kind of "wildcard" matching, which basically indicates any character otherwise mentioned in the set of all the automata: the reader is referred to Karttunen (1983) for a fuller description. Lastly, the `finality` clause indicates which states are final for the automaton given as the first argument, using the same implicit indexing as the **epenthesis** clauses: here only state 5 is non-terminal.

There are a number of linguistic implications for a finite state morphological system. First, the pair checking approach of the finite state transducer mean that the morphophonology that it encodes is necessarily "shallow". Similarly, there is no provision for the interaction of various phonological processes: the transducer either accepts a given pair or blocks at it, but there is no notion of rule ordering of the type common in generative phonological accounts.

Obviously this leads to some problems from the point of view of an accurate linguistic theory (some of which are also discussed in Karttunen (1983)). For example, the theoretical status of the symbols in the system is not always clear. Since the transducer only reads and accepts pairs of characters, more abstract phonological

[9]The indexing is actually simulated using sequential processing of the list, since standard Edinburgh syntax Prolog unfortunately does not allow random access to members of a list.

[10]Note that this is essentially rotated 90 degrees from the format used in Karttunen (1983).

processes have to be handled through the use of special, diacritic characters with little or no theoretical justification. Second, while it is theoretically possible to replace the simple symbols of the KIMMO automata with phonological matrices, the effects of underspecification on the search space is not clear.

3.1.4 An example of a finite state rule

Here we give in greater detail an illustration of the finite state transducer approach used in two-level morphological analysis. Figure 3.6, repeated here for convenience, shows the actual Prolog representation of the epenthesis automaton from Karttunen and Wittenberg (1983) as a state table. Figure 3.7 shows a network diagram, which may be easier to follow: both representation depict exactly the same information, however. In each of the diagrams, the left member of a pair of characters is the lexical form, and the right member is its surface realization. Final states are represented by double circles in figure 3.7. This automata is specifically designed to capture the alternations in spelling related to plurals: `cat` versus `cats`, `fish` versus `fishes`, `kitty` versus `kitties`, `church` versus `churches`, etc. The automaton begins in state 1, and in general continues there (⟨=.=⟩ capturing most ordinary pairs) except for pairs indicating a possible "special case". Thus, the pair ⟨c.c⟩, which may initiate an exceptional `ch` sequence, shifts the state to 2, where an immediately subsequent ⟨h.h⟩ moves the automaton to state 3. Note that the lexical representation for the plural is `+s` (see appendix A). From here, the only transition which includes a lexical + is the pair ⟨+.e⟩, which moves the automaton to state 5;

```
epenthesis ("cc", [2, 2, 2, 2, 0, 1]).
epenthesis ("hh", [1, 3, 1, 3, 0, 1]).
epenthesis ("ss", [4, 3, 3, 3, 1, 0]).
epenthesis ("SS", [3, 3, 3, 3, 0, 1]).
epenthesis ("yi", [3, 3, 3, 3, 0, 1]).
epenthesis ("+e", [0, 0, 5, 5, 0, 1]).
epenthesis ([43, 0], [1, 1, 6, 6, 0, 1]).
epenthesis ("==", [1, 1, 1, 1, 0, 1]).
finality (epenthesis, [true, true, true, true, false, true]).
```

Figure 3.6: Epenthesis automaton for English, modified slightly from Karttunen and Wittenberg (1983), p. 227.

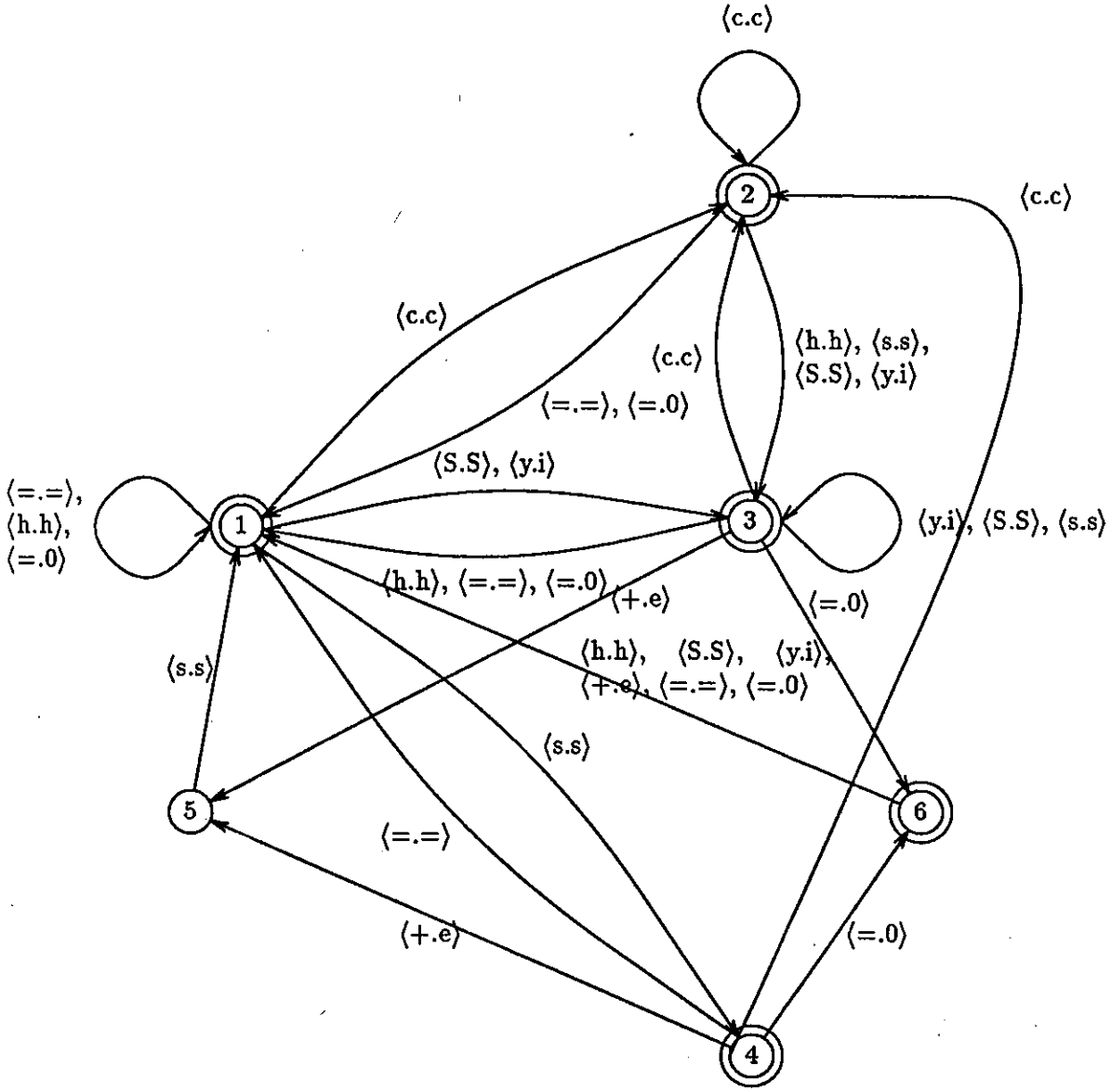


Figure 3.7: Network diagram of the epenthesis automaton.

reading the pair ⟨s.s⟩ then shifts to state 1, which (unlike state 5) can be terminal. Thus, the complete "derivation" for churches is that depicted in figure 3.8.

	(start)	c	h	u	r	c	h	+	s
		c	h	u	r	c	h	+	s
state:	1	2	3	1	1	2	3	5	1

Figure 3.8: The recognition process for the surface string churches. The upper member of a pair of characters is the lexical form. Numerals below a pair indicate the current state of the automaton *after* reading the pair.

It should be noted that the actual power of the transducers as a phonological component is a complex function of all the automata taken together. Since reading a given pair requires the satisfaction the entire set of automata, it can be difficult to determine from a global perspective whether a particular form is recognizable or not, since the process may be spread among several different automata.

3.1.5 The role of the lexicon in Pro-KIMMO

Both KIMMO and Pro-KIMMO use a multiply-branching tree structure to maximize efficiency in the process of searching the lexicon. The tree is composed of terminal and non-terminal nodes. Each node in the tree bears a symbol label and links to any daughters: if it is terminal, it also carries a list of continuation categories and syntactic features. Figure 3.9 shows a representation similar to but perhaps more intuitive than that actually used: in practice, links between parents and descendants are not explicit external links but are accomplished through nested list of complex structures, as in figure 3.10. Here the non-terminal nodes are represented by a clause with two members, the label for the node and the descendants of that node. Terminal nodes have three parameters, the second being a list of lists of continuation categories and syntactic features.

This example presented in section 3.1.4 above illustrates an important aspect of the way that Koskenniemi's system constrains the search space for correspondences. The automaton presented in figures 3.6 and 3.7 could in principle recognize a string like churchs as a morphologically simple form, *if* the lexicon were expanded to include it. It will *not* recognize such a form as a plural, however, since the lexical entry for the plural formative includes the special boundary marker +, and there is no path through the automaton which satisfies both the lexical entries and the necessary character pairings. Thus the actual constraints on possible solutions are a

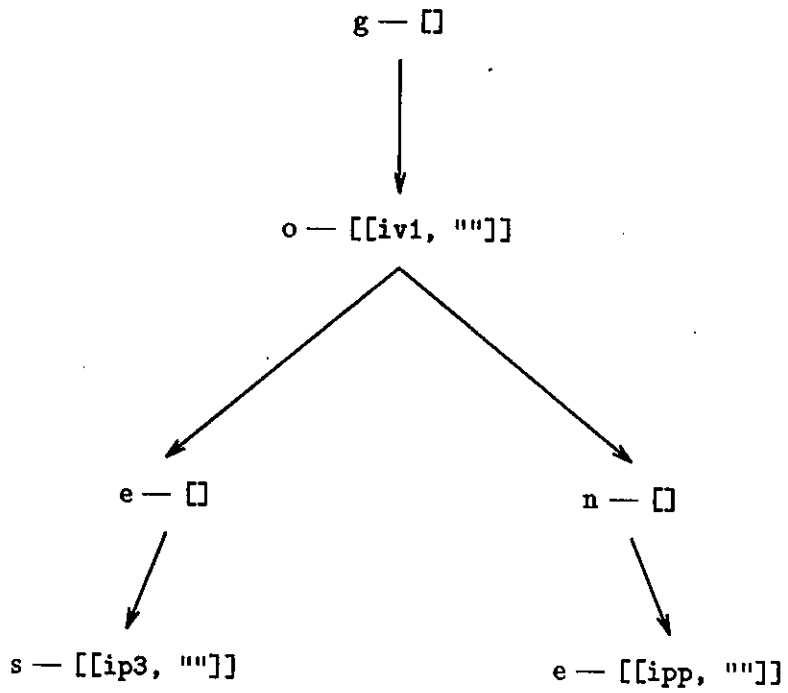


Figure 3.9: Tree representation of the lexicon

```
[(g,
    [(o, [[ iv1, ""]]
                    [ (e,
                        [(s, [[ip3, ""]], [])])
                      (n,
                        [(e, [[ipp, ""]], [])])])]
```

Figure 3.10: Nested list representation of the lexicon, pretty-printed for clarity.

result of the interaction between the lexicon, the morphosyntactic description, and the automata.

The morphosyntax of the two-level system is provided primarily through a series of continuation classes. Each lexical entry includes as its third argument a continuation class, which may be either the end of word marker #, a single class, or a special notation abbreviating several classes. Appendix C illustrates these for the Pro-KIMMO version adapted from Karttunen and Wittenberg (1983): these are essentially as originally designed, with some minor changes related to Prolog. The morphosyntactic description encodes a simple transition network: in principle, the description can be recursive, as it is for un.

3.2 Some computational considerations

3.2.1 Criteria for evaluation

In evaluating an implementation like Pro-KIMMO, there are a variety of factors, some inherently conflicting, which must be kept in mind. First, and perhaps most important, how well does this model represent the linguistic realities of morphological processing? Certainly there are a variety of competing theories of morphology, and one's choice of theory affects the design of a processing system. Koskenniemi claims both theoretical and practical advantages for the two-level model: does the theory behind KIMMO stand up to what is known about morphology in general? A complete treatment of this is beyond the scope of this paper: Karttunen (1983) mentions a few linguistic problems with the model, however.

Since Pro-KIMMO originated from a desire to implement a testable version of two-level morphology, the theoretical framework was a given: more important for this implementation was how well it actually reflected Koskenniemi's original work (particularly as implemented in KIMMO). Our aim throughout was to preserve the spirit of KIMMO as much as possible, given the very different nature of programming in Prolog.

Evaluation of a model on the basis of its efficiency is a reasonable endeavor, particularly a system which is intended for practical applications (like Koskenniemi's, which has been applied to recognizing running Finnish text (Koskenniemi (1983a))). For systems designed for educational purposes like ours, however, efficiency of operation is not as crucial. Of nearly as great importance is efficiency of implementation, in its requirements on a user or programmer's time.

3.2.2 The advantages of Prolog

One of the often-cited advantages of languages like Prolog is the degree to which one can focus on the logical details of the problem without as great a concern about control processes. Of course, there are always concerns about efficiency and preventing non-terminating searches that require some attention to ordering of clauses in the database and preventing unwanted backtracking. In general, however, the ability to concentrate on a declarative description of the problem is a significant advantage of Prolog: this proved to be beneficial in the actual implementation.

One very significant aspect of Prolog as a declarative language is the prospect of non-deterministic programming. In an appropriately specified Prolog rule, there are several ways in which the arguments of a goal term can be instantiated. The stock example of this is specifying the relationship between sublists of a complete list with a clause variously called `append` or `concatenate`. Typically, this functor takes three arguments: an initial sublist, the remaining sublist, and the whole list. The declarative description of the clause in plain English might be

> the append relationship holds between three lists if the first and second lists taken together comprise the third list,

or, written as a Prolog goal

```
?- append(Initial_list, Remainder, Whole_list).
```

What makes this technique both interesting and important is that various combinations of the three arguments can be uninstatiated (the rough equivalent of output variables in procedural languages). For example, if the goal above is called with `Initial_list = [a,b,c]` and `Remainder = [d,e,f]`, the Prolog interpreter will unify `Whole_list` with the list `[a,b,c,d,e,f]`. If, however, `Whole_list` and `Initial_list` are instantiated as above, Prolog will successfully instantiate `Remainder` to `[d,e,f]`. In fact, only instantiating `Whole_list` returns (through backtracking) various partitions of the list: `[]` and `[a,b,c,d,e,f]`, `[a]` and `[b,c,d,e,f]`, `[a,b]` and `[c,d,e,f]`, etc.

Since one of the advantages claimed for `KIMMO` is the ability to perform both generation and recognition using essentially the same mechanism, it would seem beneficial to take advantage of this aspect of Prolog's design. In fact, since our main interest was in the recognition process (which is the harder of the two), we designed the system deterministically. Some initial research indicates that the extensions necessary to do both generation and recognition with the same kind of mechanism are not major, however. This aspect of Prolog seems well suited to general research on morphological processing.

3.2.3 Some limitations of Prolog

There are also several ways in which Prolog did not perform ideally, some related to the particular implementation strategy, and others inherent weaknesses of the language. The first of these concerns the automata, discussed in section 3.1.3. While one of the real benefits of the two-level model is the efficiency of using automata to encode phonological or orthographic rules, standard Prolog does not seem well equipped to take advantage of this efficiency. The data structure described in section 3.1.3 above is handicapped by the absence of an array data type in Prolog. Since all psuedo-array processing is actually handled by sequential accessing of lists, a substantial performance penalty is paid, one which grows more severe for larger automata. This is in contrast to true matrix operations, for which it is possible to calculate an offset and move quickly to a distant cell based on its memory address.[11]

This could have been handled in at least two other ways. Perhaps the simplest is to specify each transition as an individual rule. Under this scheme, the fragment of figure 3.6 repeated here would appear as in the bottom half of the figure 3.12. This latter formulation has an important efficiency advantage over the former one, in that the facts of a particular transition are more directly available than if they must be gathered through sequential access of a list. One could even indicate that the automaton blocks on certain input by omitting the relevant facts altogether: thus, the fifth and eleventh facts above would not be strictly necessary (and would be better left out under a strictly declarative reading of the program). Unfortunately, organizing large automata in such a fashion makes it quite difficult for the user to understand exactly what they do: in this respect, a tabular arrangement is more suitable. Of course, a more sophisticated program could maintain two formats, one external and "user-friendly", and one compiled from it, designed for efficiency rather than clarity. This seems to be the approach taken by Gajek et al. (1983) in their description of the GMACHINE and RMACHINE implementations of the automata.

Another problem with `Pro-KIMMO` is the fact that Prolog has a built-in strategy of depth-first search with backtracking. Combining this with the implementation strategy of returning all possible parses (rather than the first available one) means that all possibilities are tried. Additional, there are several points of non-determinism which are dynamically expanded, in particular the approximation of phonological classes, and the set of morphological category alternations. The phonological search space is enlarged by the fact that any given character can be replaced in the automata by (in order tried)

[11] The predicate arg allows indexed access to individual arguments of a clause, but it is not well suited for true matrix operations since it also requires the arity of the predicate to be instantiated, making general purpose use awkward.

```
epenthesis ("cc", [2, 2, 2, 2, 0, 1]).
epenthesis ("hh", [1, 3, 1, 3, 0, 1]).
⋮
finality (epenthesis, [true, true, true, true, false, true]).
```

Figure 3.11: A fragment of the epenthesis automaton, repeated from figure 3.6.

```
epenthesis (1, "cc", 2).
epenthesis (2, "cc", 2).
epenthesis (3, "cc", 2).
epenthesis (4, "cc", 2).
epenthesis (5, "cc", 0).
epenthesis (6, "cc", 1).
epenthesis (1, "hh", 1).
epenthesis (2, "hh", 3).
epenthesis (3, "hh", 1).
epenthesis (4, "hh", 3).
epenthesis (5, "hh", 0).
epenthesis (6, "hh", 1).
```

Figure 3.12: The same fragment, rewritten with each state transition as an individual fact in the database.

1. itself as a literal character
2. a class symbol (with more than one possibility in some cases, e. g. in the description in Karttunen and Wittenberg (1983), s can belong to either the C class (consonants) or the S class (the class of exceptional final characters with respect to plural formation))
3. the "wildcard" symbol =

Since the automata operate on pairs of characters, there can be a large number of possibilities which must be tried in the worst case. For example, since s can fall under two different class symbols, there are four alternatives for each member of the pair, which means a total of 2^4 or 16 possibilities to test. The problem increases exponentially for more complex systems of phonological classification. Again, it appears that precompilation of the automata to fully specify in a static fashion all

literal pairs would reduce this problem, this being a step towards a full merge of the automata, as discussed in Karttunen (1983).

The same kind of problem exists for the system used to represent alternation of morphological categories. Since V functions as an abbreviation for seven continuation classes in the English description (P3 PS PP PR I AG AB), every search that reaches a node in the lexical tree which is a terminal of continuation V has to try all these possibilities. Since Pro-KIMMO implements each continuation class as a different lexical tree, this means a number of accesses of the database. This problem is compounded by the fact that this search may prove ultimately fruitless: for example, if an entry for book were added to the lexicon listed in appendix A, an attempt at recognizing it would have to go through all the verb continuations because the verb boo happens to occur as an initial substring of it. This problem becomes especially severe for the Pro-KIMMO implementation of the description of Japanese in Alam (1983), since there are up to 12 alternations for each category. This complexity combined with generally longer strings pushes the recognition time for Japanese tokens like kattemita up over a minute for Pro-KIMMO: this seems unacceptably long for any serious system.

The fundamental problem here is that all solutions are returned, hence the entire solution space must be searched exhaustively. This renders the relative merits of depth-first or breadth-first search strategies irrelevant. Combined with dynamic exploration of a number of points of non-determinism (for example, the specification of character pairs), and the indiscriminate nature of automatic backtracking in Prolog, it is hardly surprising that Pro-KIMMO is highly inefficient. This is only compounded by the problems concerning operating the automata described above. Of course, Prolog is not necessarily the language of choice for highly efficient execution of programs: its strongest advantages lie elsewhere. More effort at optimizing the performance of the model could presumably decrease execution time substantially, though probably at the expense of some simplicity and clarity in the code.

3.3 Areas for future work

There are a number of features of the two-level model which could bear extension, revision, or replacement. First of all, the phonological representation used in current models is sparse in the extreme: there is no adequate provision for distinctive features, tiers, or other theoretical devices of importance to current phonological research. Clearly, the two-level model needs substantial revision here if it is to keep up with linguistic theory.

Moving in this direction has the consequence of making the phonological com-

ponent more and more complex, with the result that it will become more and more time-consuming to operate. In theory, there may come a point at which the simultaneous satisfaction of lexicon, morphological, and phonological constraints will prove too costly for exhaustive search. For example, at present the full phonological machinery must be satisfied even for searches that will ultimately fail for lexical or morphosyntactic reasons. It may prove more desirable to operate a simplified phonological machine on a first pass, and then to test surviving candidates with the full phonology. [12] How well this strategy might work is a subject for empirical verification.

A different issue concerns the use of finite-state machinery in the two-level model. While a finite-state approach to morphology presents a more restricted theory than other, more powerful grammar types (notably context-free systems), it is not clear whether it is actually powerful enough to usefully represent the full scope of morphological relations in natural language. In particular, dependencies between (possible non-adjacent) formatives pose a real problem for a finite-state grammar. The general treatment of these in a two-level system can take one of two different approaches, neither of them satisfactory from a linguistic standpoint. One approach is subdivision of the lexicon: this has the undesirable effect of requiring duplication of entries, particularly if the elements involved allow intervening formatives. The other method is use of diacritic characters to mark co-occurrence restrictions, along with transducers to filter out ill-formed strings. There is certainly no linguistic justification for such diacritics, however, other than making a particular analysis come out right: this amounts to nothing more than "smuggling" the morphosyntax into the phonology. The fundamental issue is whether use of context-free phrase structure rules would provide for more suitable descriptions: we expect to pursue this possibility in future work.

[12] As a first approximation, this might mean allowing permissible structural changes without requiring the satisfaction of structural descriptions. Expressing this in a principled way with KIMMO-style automata would be difficult, however.

Appendix A: The English lexicon for Pro-KIMMO

This is straightforwardly a Prolog implementation of the LISP version found in Karttunen and Wittenberg (1983), with only one or two changes (to correct presumed errors). Each clause consists of:

- a class label, indicating the category of the entry
- a character string, where [0] represents the empty string
- a continuation class (# indicates that it is terminal, i. e. no continuation is allowed)
- a feature description (possibly empty)

The lexicon presented in Karttunen and Wittenberg (1983) further distinguishes continuation classes which begin with a slash as continuable: since this seemed to overlap completely with the third argument of the lexical entry, it was omitted.

This is the form of the lexicon that the user develops: as in KIMMO, it is then compiled by the program into a letter tree for more efficient search.

```
lexicon (n, [0], c1, "N SG").
lexicon (n, "+s", c2, "N PL").
lexicon (mn, [0], c1, "MASS N"). % Kartunnen & Wittenburg say c2: a typo?
lexicon (c1, [0], #, "").
lexicon (c1, "'s", #, "GEN").
lexicon (c2, [0], #, "").
lexicon (c2, "'", #, "GEN").
lexicon (p3, "+s", #, "V PRES SG 3RD").
lexicon (ip3, [0], #, "V PRES 3RD SING").
lexicon (ps, "+ed", #, "V PAST").
lexicon (ips, [0], #, "V PAST").
lexicon (pp, "+ed", #, "V PAST PRT").
lexicon (ipp, [0], #, "V PAST PRT").
lexicon (pr, "+ing", #, "V PROG").
lexicon (i, [0], #, "V").
lexicon (ip1, [0], #, "V PRES SING 1ST").
lexicon (ag, "+er", n, "AG").
lexicon (pa, [0], #, "A").
lexicon (ca, "+er", #, "A COMP").
lexicon (cs, "+est", #, "A SUP").
lexicon (ly, "ly", #, "ADV").
lexicon (ab, "+able", #, "VERB ABL").
lexicon (root, "am", #, "V PRES SING 1ST").
lexicon (root, "are", #, "AUX").
lexicon (root, "are", #, "V PRES PL").
```

```
lexicon (root, "are", #, "V PRES SING 2ND").
lexicon (root, "at", #, "PREP").
lexicon (root, "at'tack", n, "").
lexicon (root, "at'tack", v, "").
lexicon (root, "be", #, "AUX").
lexicon (root, "be", iv1, "").
lexicon (root, "beer", n, "").
lexicon (root, "believe", v, "").
lexicon (root, "big", a, "").
lexicon (root, "bit", ips, "").
lexicon (root, "bite", iv2, "").
lexicon (root, "bitten", ipp, "").
lexicon (root, "boo", v, "").
lexicon (root, "boy", n, "").
lexicon (root, "cacti", in, "N PL").
lexicon (root, "cactus", in, "N SG").
lexicon (root, "cat", n, "").
lexicon (root, "church", n, "").
lexicon (root, "cool", a, "").
lexicon (root, "day", n, "").
lexicon (root, "did", ips, "").
lexicon (root, "die", v, "").
lexicon (root, "do", iv1, "").
lexicon (root, "does", ip3, "").
lexicon (root, "done", ipp, "").
lexicon (root, "fox", n, "").
lexicon (root, "go", iv1, "").
lexicon (root, "goes", ip3, "").
lexicon (root, "gone", ipp, "").
lexicon (root, "grouch", n, "").
lexicon (root, "had", ips, "").
lexicon (root, "had", #, "AUX").
lexicon (root, "has", ip3, "").
lexicon (root, "has", #, "AUX").
lexicon (root, "have", #, "AUX").
lexicon (root, "have", iv1, "").
lexicon (root, "ice", mn, "").
lexicon (root, "industry", n, "").
lexicon (root, "is", ip3, "").
lexicon (root, "kill", v, "").
lexicon (root, "kiss", n, "").
lexicon (root, "kiss", v, "").
lexicon (root, "mice", in, "N PL").
lexicon (root, "milk", mn, "").
lexicon (root, "mouse", in, "N SG").
lexicon (root, "oc'cur", v, "").
lexicon (root, "race", n, "").
```

```
lexicon (root, "race", v, "").
lexicon (root, "rally", n, "").
lexicon (root, "refer'ee", n, "").
lexicon (root, "refer'ee", v, "").
lexicon (root, "re'fer", v, "").
lexicon (root, "ski", n, "").
lexicon (root, "sleep", iv2, "").
lexicon (root, "slept", ipp, "").
lexicon (root, "slept", ips, "").
lexicon (root, "spy", n, "").
lexicon (root, "spy", v, "").
lexicon (root, "tie", v, "").
lexicon (root, "tiptoe", v, "").
lexicon (root, "toe", n, "").
lexicon (root, "travel", n, "").
lexicon (root, "travel", v, "").
lexicon (root, "try", v, "").
lexicon (root, "un", root, "NEG").
lexicon (root, "under'stand", iv2, "").
lexicon (root, "under'stood", ipp, "").
lexicon (root, "under'stood", ips, "").
lexicon (root, "undid", ips, "").
lexicon (root, "undo", iv1, "").
lexicon (root, "undoes", ip3, "").
lexicon (root, "undone", ipp, "").
lexicon (root, "untie", v, "").
lexicon (root, "went", ipp, "). % Kartunnen & Wittenburg (1983) have ips
lexicon (root, "went", ips, "). % for both of these: must be a typo?
```

Appendix B: The Pro-KIMMO automata

There are six automata in all: elision, epenthesis, gemination, i_spelling, surface, and y_spelling.

```
finality (elision,[true,true,true,false,true,false,false,false,false,
                   false,true,false,false,true,true]).
elision ("VV",[2,1,1,0,1,0,1,1,0,0,1,0,0,1,1]).
elision ("ii",[2,1,1,0,1,0,1,1,0,0,1,0,1,1,0]).
elision ("ee",[3,5,5,0,1,0,0,1,0,1,14,0,1,1,0]).
elision ([101,0],[4,6,6,0,4,0,0,0,0,0,12,0,0,0,0]).
elision ([43,0], [1,1,9,8,7,10,0,0,0,0,1,13,0,15,1]).
elision ("gg",[11,11,11,0,11,0,1,0,1,0,1,0,0,11,11]).
elision ("cc",[11,11,11,0,11,0,1,0,1,0,1,0,0,11,11]).
elision ("bb",[5,1,5,0,1,0,1,0,1,0,1,0,0,1,1]).
elision ("==",  [1,1,1,0,1,0,1,0,1,0,1,0,0,1,1]).
```

```
finality (epenthesis,[true,true,true,true,false,true]).
epenthesis ("cc",[2,2,2,2,0,1]).
epenthesis ("hh",[1,3,1,3,0,1]).
epenthesis ("ss",[4,3,3,3,1,0]).
epenthesis ("SS",[3,3,3,3,0,1]).
epenthesis ("yi",[3,3,3,3,0,1]).
epenthesis ("+e",[0,0,5,5,0,1]).
epenthesis ([43,0],[1,1,6,6,0,1]).
epenthesis ("==",[1,1,1,1,0,1]).

finality (gemination,[true,false,true,true,true,true,true,true,true,true,
   true,true,true,true,true]).
gemination ("VV",[4,1,0,16,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("bb",[1,0,0,5,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("dd",[1,0,0,6,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("ff",[1,0,0,7,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("gg",[1,0,0,8,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("ll",[1,0,0,9,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("mm",[1,0,0,10,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("nn",[1,0,0,11,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("pp",[1,0,0,12,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("rr",[1,0,0,13,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("ss",[1,0,1,14,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("tt",[1,0,0,15,16,16,16,16,16,16,16,16,16,16,16,16]).
gemination ("+b",[0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0]).
gemination ("+d",[0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0]).
gemination ("+f",[0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0]).
gemination ("+g",[0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0]).
gemination ("+l",[0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0]).
gemination ("+m",[0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0]).
gemination ("+n",[0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0]).
gemination ("+p",[0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0]).
gemination ("+r",[0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0]).
gemination ("+s",[0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0]).
gemination ("+t",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0]).
gemination ([43,0],[1,0,0,1,3,3,3,3,3,3,3,3,3,3,3,16]).
gemination ([96,0],[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]).
gemination ("==",[1,0,0,16,1,1,1,1,1,1,1,1,1,1,1,16]).

finality (i_spelling,[true,false,false,false,true,true,false]).
i_spelling ("iy",[2,0,0,0,1,0,0]).
i_spelling ([101,0],[1,3,0,0,1,1,0]).
i_spelling ([43,0], [1,0,4,0,1,7,0]).
i_spelling ("ii",[5,0,0,1,1,0,0]).
i_spelling ("ee",[1,0,0,0,6,0,0]).
i_spelling ("==",  [1,0,0,0,1,1,1]).
```

```
finality (surface,[true]).
surface ([0,0],[1]).   % needed for revealing 0 morphs at end of word
surface ("aa",[1]).
surface ("bb",[1]).
surface ("cc",[1]).
surface ("dd",[1]).
surface ("ee",[1]).
surface ("ff",[1]).
surface ("gg",[1]).
surface ("hh",[1]).
surface ("ii",[1]).
surface ("jj",[1]).
surface ("kk",[1]).
surface ("ll",[1]).
surface ("mm",[1]).
surface ("nn",[1]).
surface ("oo",[1]).
surface ("pp",[1]).
surface ("qq",[1]).
surface ("rr",[1]).
surface ("ss",[1]).
surface ("tt",[1]).
surface ("uu",[1]).
surface ("vv",[1]).
surface ("ww",[1]).
surface ("xx",[1]).
surface ("yy",[1]).
surface ("zz",[1]).
surface ("''",[1]).
% this is included here for the predicate that makes the list of concrete
% pairs: this is somewhat misleading,since it isn't really allowable
% on the surface
surface ("==",[1]).

% obviously this automata could be expressed more succinctly and
% declaratively with something like
%     surface ([Char,Char],[1]) :-
%         alphabet(Char).
% and an appropriate delineation of the allowable alphabetic characters


finality (y_spelling,[true,true,false,false,true,false]).
y_spelling ("CC",[2,2,0,1,1,0]).
y_spelling ("yy",[1,5,0,1,1,0]).
y_spelling ("yi",[0,3,0,0,0,0]).
y_spelling ("+=",[1,1,4,1,6,0]).
```

```
y_spelling ("ii",[1,1,0,0,1,1]).
y_spelling ("aa",[1,1,0,0,1,1]).
y_spelling ("==",[1,1,0,1,1,0]).
```

Appendix C: Morphosyntactic alternations

These represent the possible expansion(s) of the lexical continuation classes. Note that the actual continuation for a given formative, and hence its real morphosyntax, is specified in its lexical entry.

The `alternation` relation below is processed by a test of membership in the list which comprises the second argument. It could be described declaratively as

> $\mathcal{A}$ is a true alternation of $\mathcal{C}$ if there exists a fact `alternation(`$\mathcal{C}$`,`$\mathcal{L}$`)` and `member(`$\mathcal{A}$`,`$\mathcal{L}$`)` is true.

Other information is used for auxiliary purposes. The categories used are exactly those of Karttunen and Wittenberg (1983): they could be improved somewhat by more mnemonic (or less terse) labels.

```
% this initial information is for error checking and the generation routines
categories ([n, mn, c1, c2, p3, ip3, ps, ips, pp, ipp, pr, i, ip1, ag,
pa, ca, cs, ly, ab, root, #]).

continuable ([n, in, mn, v, iv1, iv2, a]).

% the abbreviatory/idiosyncratic alternations
alternation (in, [c1]) :- !.
alternation (v, [p3, ps, pp, pr, i, ag, ab]) :- !.
alternation (iv1, [pr, i, ag, ab]) :- !.
alternation (iv2, [p3, pr, i, ag, ab]) :- !.
alternation (a, [pa, ca, cs, ly]) :- !.
alternation (#, []) :- !.
% the default case: a category "alternates" only with itself
alternation (Cat, [Cat]).
```

Chapter 4

IceParse: A Model Of Inflectional Parsing and Word Recognition For Icelandic Ablauting Verbs

Thomas L. Cornell

4.1 Background

Among the first researchers to become interested in bringing the results of current morphological theory (cf. Chomsky (1970), Halle (1973), Aronoff (1976)) to bear on the task of analyzing word structure on-line were Koskenniemi (1983b), for Finnish, and Hankamer (1986), for Turkish. Their work, which I will refer to as Finite-State Morphology, while different in many details, shares a similar architecture. Morphotactics is modeled as a finite state transition network, with each new morpheme sending the parser into a new state, from which only a limited set of subsequent states (i.e. morphemes) can be reached. Allomorphy is handled by rules, when those rules can function given only surface and underlying forms as input,[1] and by the affix lexicon, when alternations are not recoverable from the surface. Both parsers use the same basic generate-and-test, or analysis-by-synthesis parsing strategy, which involves hypothesizing that each lexical root may underlie the input form, and then

[1]The phonology too is limited to a statement of this surface/lexical correspondence.

attempting to derive the input form from this candidate solution.[2]

I will attempt to show in what follows that the FSM paradigm as it is currently construed lacks descriptive adequacy in the face of the world's morphologies. Furthermore, I will show that it misses generalizations even within its coverage. I will also suggest that FSM is incompatible with recent psycholinguistic results. I will propose another model which answers these problems, though in other respects its debts to the FSM paradigm are manifold. Finally, I will develop a formal notation in which autosegmental representations can be written on a tape which a simple automaton can scan one move at a time, thus allowing morphological parsers to bring to bear the advances in phonology of the last decade on the problem of parsing words.

The descriptive problems of FSM stem from its strict adherence to Item-and-Arrangement ("beads-on-a-string") morphology. Thus morphemes are indivisible substrings of the word, and morphological parsing essentially amounts to inserting hyphens in the right places and providing a morpheme-by-morpheme interlinear gloss.[3] Some common morphological operations cannot be parsed in this way, however. Consider the processes of Ablaut (as in English *sing-sang-sung*) and Umlaut (as in English *mouse-mice*), which operate on stem vowels, leaving the consonantal skeleton intact. This state of affairs bears some surface similarities to templatic morphologies, like Semitic (cf. McCarthy (1981)). Thus we might claim that Ablaut and Umlaut constitute the association of single-vowel melodies with roots like /*sŋ*/ or /*ms*/, and so adopt a device like Kay (1985)'s parser for CV-morphology. In his model, Classical Arabic roots, vowel melodies and CV-skeleta each have their own lexical entries—the surface word is composed of the superposition of three strings. Few people would argue, however, that with no discernible templatic regularities, a vowel melody of (always and only) a single vowel, and no apparent spreading effects, a system could reasonably be claimed to be an instance of templatic morphology.

It gets worse. In the Icelandic Strong Verb paradigms Ablaut and Umlaut interact, so that we might get a derivation like that in (1).

[2]More recent examples of FSM parsers include Kay (1985)'s parser for Arabic, which follows the Two-Level model of Koskenniemi, and Cole (1986)'s parser for Finnish, which follows Hankamer's model. These newer FSM parsers incorporate the rules and representations of CV morphology and phonology. Specifically, Kay (1985) is based on the work of McCarthy (1981) and Cole (1986) is based on the analysis of Finnish by Keyser and Kiparsky (1984).

[3]Kay's parser presents only an apparent exception to this adherence to Item-and-Arrangement morphology. However, even though his parser (see fn. 2) is for classically non-concatenative morphology, it still upholds the indivisibility and atomicity of the morpheme. The "hyphens" of this model, as it were, are vertical rather than horizontal.

(1) /brest-/ '(to) burst'

brust- < Ablaut (=Preterite stem)
bryst- < Umlaut (=Preterite Subjunctive stem)

Even in radical autosegmental terms, we cannot claim that Ablaut represents a melody on one feature-tier and Umlaut a melody on another.[4] Clearly what is needed is some process-based account of morphology, along the lines of Anderson (1987) (cf. also Thomas-Flinders (1981), Matthews (1972)), which captures morphotactics and allomorphy by means of morphological rules. Thus we might capture the derivation in (1) with the rules in (2).

(2) (a) Preterite Stem Ablaut

$$[-cons] \longrightarrow u \,/\, \left[\overline{} \atop +Preterite \right]^{5}$$

(b) Preterite Subjunctive Umlaut

$$[-cons] \longrightarrow [-back] \,/\, \left[\begin{matrix} \overline{} \\ +Preterite \\ +Subjunctive \end{matrix} \right]$$

This same solution also helps us around FSM's second shortcoming, which concerns the area of morphotactics. According to FSM, each morpheme is lexically specified for a continuation category (computationally equivalent to a new state of a finite-state automaton) which includes all and only the morphemes which may follow the current one. Consider, however, the case of Icelandic, where the 1stPl. suffix *-um* may attach to any stem, but where the 2ndPl. is marked by *-ið* in the Present tense, and by *-uð* in the Preterite tense. Thus the continuation category for the Present stem is {*-um, -ið*, ...}, and that for the Preterite stem is {*-um, -uð*, ...}. The distribution of *-um* is implicit in this system, and can be recovered by carefully intersecting all the relevant continuation categories, but we might hope for some more explicit characterization of a given affix's distribution.

[4]Historically this might have been a possible analysis: Ablaut was originally an alternation in lowness between d and a, and Umlaut was an alternation in backness. However, it is not at all certain that Umlaut had morphological significance at any time when the lowness alternation of PIE Ablaut was still recoverable.

[5]Actually this rule conflates two rules. Zero-grade Ablaut deletes the stem vowel, leaving the off-glide, if there is one, or triggering u-Epenthesis, if there is no glide. Example (1) represents this latter case.

Once again the notion of morphemes-as-rules comes to our rescue. We can capture the distributions of each morphological operation explicitly, as in example (3) below.[6]

(3) Some Icelandic Suffixing Rules

$$(a)\quad X \rightarrow X + um \;\Big/ \begin{bmatrix} +Plural \\ +1st \end{bmatrix}_X$$

$$(b)\quad X \rightarrow X + i\eth \;\Big/ \begin{bmatrix} -Preterite \\ +Plural \\ +2nd \end{bmatrix}_X$$

$$(c)\quad X \rightarrow X + u\eth \;\Big/ \begin{bmatrix} +Preterite \\ +Plural \\ +2nd \end{bmatrix}_X$$

Finally, the psycholinguistic research of Emmorey (1987) has shown that, while even bound roots like *-mit* and *-ceive* show facilitation effects in priming experiments, no similar effects can be found for suffixes. This leads to the conclusion that, unlike stems (even bound forms), suffixes do not have entries in the mental lexicon. This is compatible with the morphemes-as-rules model, wherein suffixes (i.e. suffixation rules) are qualitatively different formal entities as compared to lexical items. Of course, these results are also compatible with a model where each form in the paradigm has a separate entry, all associated with a single lexeme (cf. Wehrli (1985)). Crucially, however, this result is not compatible with FSM.

4.2 IceParse

The model I propose (which I will refer to as IceParse in what follows, since its planned implementation is for Icelandic inflectional morphology) is based on the Cohort Model of spoken word recognition (Marslen-Wilson (1980) and references

[6]This instance is in some respects similar to the problem of Discontinuous Dependencies, whereby morphotactic restrictions must make themselves felt across several intervening morphs (e.g. in the abstract situation defined by such strings of "morphemes" as A-Y-Z-B, C-Y-Z-D, but *A-Y-Z-D or *C-Y-Z-B). In a finite-state system this frequently requires that whole sub-parts of the transition network be repeated.

cited there), as extended to morphology by Emmorey (1987), with the additional requirement that the same device also return a morphosyntactic representation (cf. Anderson (1987)) of the word. The idea behind the Cohort Model is that all lexical items compatible with the input heard so far are considered part of a "cohort" of active lexical items. As more input is heard, the number of items in the cohort drops, until only one item remains active. The point at which the cohort is reduced to only one member (the "uniqueness point") may occur before all the input has been processed; this model thus accounts for why people appear to be able to recognize a word before they have heard all of it. The crucial fact to note for our purposes is that human beings appear to process input as it comes in, in the order in which it is presented.

Note that Marslen-Wilson does not give an account of phonological and morphological processing anywhere in the Cohort Model. If this model is to be augmented with morphological and phonological constraints on cohort membership, the following constraint seems to be required: the rules so invoked cannot hold up processing while they wait for future context to confirm their analyses of the input. That is, they cannot backtrack. This in turn suggests that humans use a parallel processing strategy (cf. Dolan (this volume)) under which they are able to maintain multiple parses in working memory at the same time. If we consider these parallel parses to represent one underspecified, ambiguous parse, we begin to see the parsing problem as essentially a problem in the management of ambiguity.

In order to manage the ambiguity which this necessarily non-deterministic process introduces, the parser needs to have access to some stored knowledge so as to constrain what it may expect a possible analysis of the input to look like. The most pertinent source for such knowledge I take to be the morphology and phonology of the language. Thus if the parser initially assumes that the input contains a valid word of the language, this constrains possible analyses of the input to those which yield valid intermediate representations, and a valid underlying form. This in turn suggests a very concrete role for the grammar and the notion of grammaticality in a processing system—it acts as a constraint on the tolerable ambiguity of the system. Hence ungrammaticality represents in parsing terms an increase in ambiguity beyond some optimal, but not necessarily critical threshold. This must be the case since we know that many ungrammatical utterances are still interpretable, and as often as not partial structures can be assigned to them. Presumably then the parser does not simply halt if it detects an ungrammatical string, but rather shrugs and continues onwards, assigning the input some distorted, incomplete representation. Other modules (especially the lexicon, in this case) so constrain the analysis that ideally only a small set of analyses remain active at the end of a distorted parse, which can be narrowed down further by syntactic and semantic context.

4.3 Rules

Computationally, let us imagine that the (inflectional) parser is a function $\mathcal{P}$ which maps an input string of some alphabet to an ordered pair consisting of a string in the lexicon and a set of syntactic features. The most straightforward way to bring grammatical knowledge to bear on the parsing problem is to break $\mathcal{P}$ down into a sequence of smaller functions (p_1, p_2, ... p_n) each of which corresponds to one rule of a morphological and phonological derivation, run in reverse. Thus a generative rule of the form A → B / C — D becomes (roughly) B → A / C — D.[7]

We model the functions p_i involved in this reverse derivation as automata (called "transducers") which scan morphological and phonological representations and write out interpretations of those representations, which in turn serve as input to the next rule/function p_{i+1}. The control of a given automaton is represented as a finite state transition network.[8]

To get an idea of how this works for simple strings, consider the rule of u-Umlaut given in (4).

(4) Icelandic u-Umlaut

$$a \rightarrow 0 \Big/ \_\_\_\_ C^* \begin{bmatrix} +high \\ +round \end{bmatrix}, \quad \text{where C is [+cons].}$$

The action of undoing this rule can be modeled by the transducer whose control function is depicted in Figure 4.1. States are numbered, "final" states (explained below) are drawn with a double circle. 'C' stands for [+cons] segments (as in the rule above), 'u' in this case is taken to symbolize all [+high, +round] segments, '–u' stands for all the segments which are not 'u', and '=' as the input symbol stands for "anything at all," and as the output symbol for "the same thing as in the input."[9]

The transducer works as follows. Every arc of the transition network is labeled with an (input.output) pair. A move of the transducer consists of looking at the current input symbol and the current state of the controller, and determining which arc out of the current state is labeled with the current input. The transducer then enters the state to which that arc points, and reads the next input symbol. The

[7]Of course, the possibility of an underlying B must typically be considered as well.

[8]Note that nothing hinges on the finite-stateness of these transition networks. As it happens, finite state nets are sufficient for the tasks discussed here, so there is simply no reason to adopt some more powerful device. (However, see Johnson (1972) for an argument that SPE-style phonological rules can in fact be restricted to finite-state power.)

[9]Computationally, the use of variables like '=' or '–u' is equivalent to drawing a fan of arcs from, e.g., state 2 to state 1, one arc for each possible pair of identical segments except for the pair (u.u).

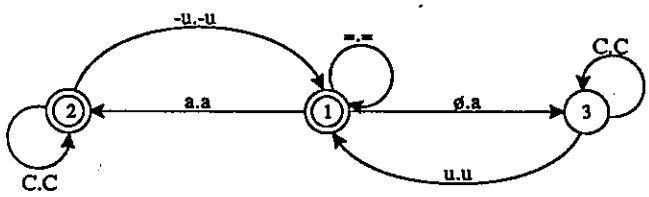


Figure 4.1: u-Umlaut Automaton

rules of automata theory state that there must be a well-defined transition available for each successive input symbol, and that when the end of the input string has been reached, the machine must be in a designated final state. If either of these conditions is not met, the string is not a string of the language and no proper representation can be assigned to it by a transducer.

For example, if the u-Umlaut device reads an *a*, it enters state 2. State 2 guarantees that an input *a* will not be followed by a [+high, +round] segment. This is so because there is no well-defined transition out of state 2 labeled *u*. If, on the other hand, a 0 is encountered in the input, and analyzed as resulting from an underlying *a*, state 3 is entered, requiring that the following [–cons] segment be [+high, +round]. If no [–cons] segment at all follows, then the transducer will finish scanning the input still in state 3, and will therefore end in a non-final state. Note that the alternative parse, whereby a surface 0 corresponds to an underlying 0, will not cause the transducer ever to leave state 1.

Morphological rules function similarly, except that, in addition to information about the phonological form, they have access to a set of syntactic features which also condition the rule's operation. Icelandic presents two types of morphological operation: suffixation and alteration of the stem vowel. This latter type (which includes Ablaut and Umlaut) works exactly like a phonological rule, except for the presence of syntactic features in its Structural Description. Consider the following case of an Ablaut rule.

(5) Preterite Singular Ablaut

$$e \rightarrow a \,/\, \left[\begin{array}{c} \overline{\qquad} \\ -Subjunctive \\ +Preterite \\ -Plural \end{array}\right]$$

e.g. *gaf* '(I) gave'
but *gef* '(I) give'

Since the syntactic features associated with a word are associated with the whole form, they are presumably available to any arc(s) which needs to consult them. Thus we might have the fragment of a control diagram given in Figure 4.2 for Preterite Singular Ablaut.

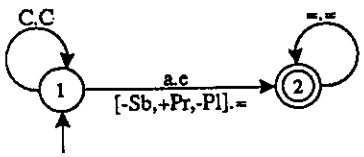


Figure 4.2: Preterite Singular Ablaut Automaton

Suffixation rules are basically the same, but since they introduce phonological material into the form, they operate like insertion rules. Thus the rule for suffixing the Preterite 2Sg. ending *-st*, given in (6), might correspond to the automaton fragment pictured in Figure 4.3 (where C and V refer to values of [syllabic]).

(6) Preterite 2Sg. Suffix Rule

$$(a) \quad X \rightarrow X + st \,/\, \left[\begin{array}{c} -Subjunctive \\ +Preterite \\ -Plural \\ +You \end{array}\right]_X \text{——}$$

e.g. *gafst* '(you) gave'
but *gefur* '(you) give'

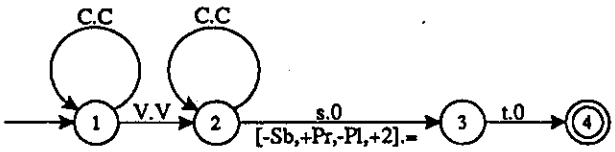


Figure 4.3: Preterite 2Sg. Suffix Automaton

This automaton introduces a new device. The '0' which appears in the output labels causes no underlying phonological segment to be written to the output. In essence it is an instruction to the parser simply to consume a symbol of input.

Though the formal device whereby we represent syntactic features as labels on arcs is the same as for phonological symbols, the parser must treat syntactic features rather differently, since they do not appear in the parser's input. In order for it to jump an arc, a given transducer's input must match the label, where by match we mean it must not clash in any feature value. This unification procedure, however, leaves open the possibility that either the input or the label might be unspecified for some feature, in which case whichever of the input or the label has an explicit value for a feature will "win out," and assign that value to the proposed output representation. Thus, in the case of the *-st* Suffix automaton, a relatively "shallow" rule, the syntactic label of an arc will always match, so if the input contains an *s* at the relevant point, this will cause the feature set [–Subjunctive, +Preterite, –Plural, +You] to be written to the output.

Another difference in the handling of syntactic features is their treatment by the bottom-most rule, p_n. Since the inflectional forms of a word are not present in the lexicon (except for exceptional forms), the syntactic features associated with the word are not constrained by the lexicon but rather by the morphosyntactic representation on the relevant leaf node of the syntactic phrase marker. Ideally, a full-scale Natural Language Processing system would have placed some constraints on this morphosyntactic representation based on properties of the phrase marker as parsed to that point; however, IceParse is just a morphological parser and has no context larger than the domain of the word. Thus the morphosyntactic representation is entirely built up by the output of p_n. However, as we will see below, once p_n has written something, that constrains future possiblilities (thus modeling the constraints of the morphotactics).

Note before we go on the characteristic of morphological paradigms that they

typically represent the effects of disjunctive blocks of rules (Anderson (1986a)). Thus, in Icelandic, if one Ablaut rule fires, no other Ablaut rule may fire, and if one agreement suffixation rule fires, no other agreement suffixation rule may fire (up to ambiguity). It is thus a waste of resources to run the same form through each Ablaut transducer, since only one may apply, and we will normally combine such disjunctive blocks of transducers into one large one. Accordingly, the full control functions for Ablaut and AGR-Affixation are given in Figure 4.4 and Figure 4.5 (disregarding here the non-finite forms, and the processing of irrelevant parts of the word). Preterite Singular Ablaut is handled by states 1, 4, and 5 of the Ablaut Automaton, and *-st* Suffixation is handled by states 1, 2, and 3 of the AGR-Affixation Automaton.

The actual functioning of the transducers is somewhat more complicated than the u-Umlaut example would suggest, since they must be able to run parallel parses simultaneously. What happens, roughly, is that the automaton looks at its currently available arcs, and proposes those which match its input as live possibilities. In other words, a given transducer may have available to it at a given moment several analyses which might correspond to the input. It is able to write all of these candidate solutions to its output, in a sense using the set of these proposed analyses to represent an ambiguous parse. The output labels on these proposed arcs in turn serve as input to the next transducer in the cascade which does the same thing. Thus, not only might a transducer produce multiple outputs, but it may also receive multiple inputs. The set of proposed live arcs must comprise all those arcs compatible with any input. The potential for combinatorial explosion is restrained by (a) the Lexicon (see the next paragraph), (b) the fact that activating any state of a rule sets up expectations which may not be fulfilled–thus parses are killed off as new ones are generated, and (c) the fact that many of a rule's inputs may be collapsed into variables, since the set of symbols a given rule is sensitive to is typically small.

Finally, the proposals of the bottom-most rule are checked against the lexicon. The lexicon, in accordance with Marslen-Wilson's Cohort Model, is structured like a tree, whose nodes are segments and paths through which correspond to words. Terminal nodes mark uniqueness points. The tree corresponds to a cohort of active lexical items (given strict left-to-right processing) in that the members of the cohort correspond to those terminal nodes of the tree which are dominated by the current node (the node which represents the underlying form of the current input symbol). The bottom-most rule (p_n) is constrained by the Lexicon to jump only those of its arcs whose output label corresponds to a daughter of the current lexical node. Those arcs propsed in the output of p_n which do not result in a step down the lexical tree are *killed off* by being removed from the set of p_n's proposed analyses—

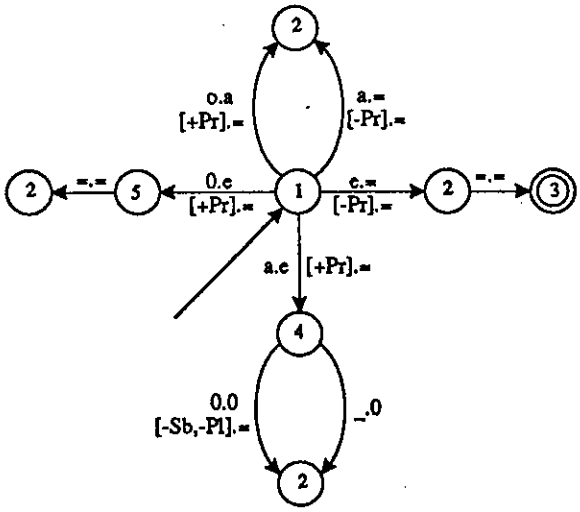


Figure 4.4: Ablaut Automaton

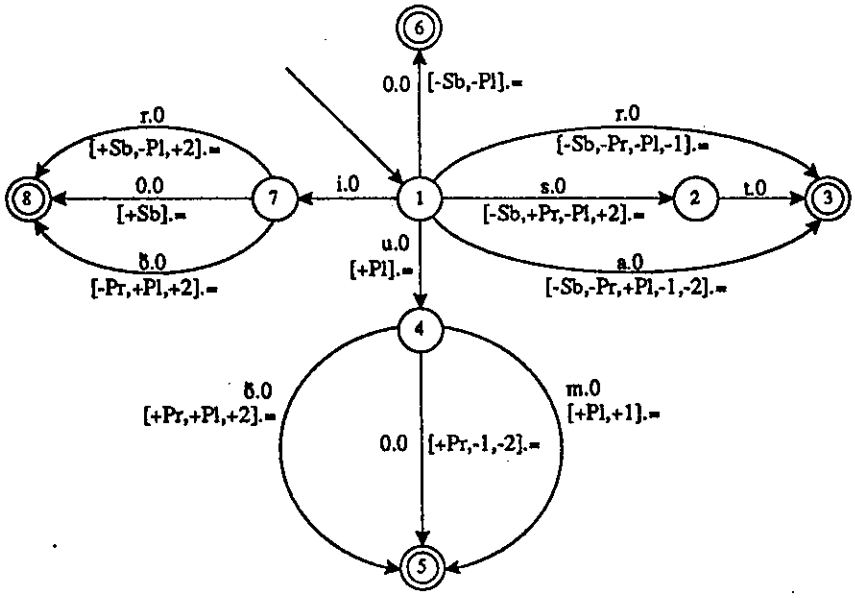


Figure 4.5: AGR-Affixation Automaton

this makes the states they lead to *inaccessible*. P_n is now allowed to enter any states which can still be reached by live arcs. It in turn kills off any of its inputs (the output set of p_{n-1}) which do not lead to (live) transitions in its own control function, allowing p_{n-1} to enter any states still accessible across live arcs, and so on up the line. Overall, what happens is that a rule proposes plausible analyses and then, before actually adopting any of them, receives feedback from the lexicon and other rules as to which of those outputs are actually possible, given the knowledge contained in the grammar and the input so far.[10]

To see how this works, and how morphotactics are implemented in $\mathcal{P}$, consider how the parser would deal with the word *gafst*, Preterite Indicative 2Sg of *gefa*, 'to give.' In this case, Ablaut plays the role of p_n. Since Affixation is the shallower rule it will apply first in the parse, but it is set up to loop back to state 1 (arcs not shown) until the stem syllable nucleus has been scanned. Thus, at first, it will take a totally unspecified input morphosyntactic representation and it will output the same totally unspecified representation. Ablaut, in this two-rule system, will see unaltered input. It in its turn is set up to consume the stem onset (arc not shown), remaining in state 1 until the first vowel is read. The stem vowel, *a*, causes two arcs to be proposed: ${}_1(\text{a.=})_2$ and ${}_1(\text{a.e})_4$.[11] Since there are no verbs in Icelandic which begin *ga-*,[12] the arc ${}_1(\text{a.=})_2$ will be killed off, leaving only state 4 accessible on return of control from the lexicon. The Ablaut automaton can enter state 4 because it has no syntactic information which conflicts with the syntactic label on ${}_1(\text{a.=})_2$, indeed it has no syntactic information at all, so far.

On the next move of the parser, the Affixation Automaton will simply consume the *f* of *gafst*, passing it through to the Ablaut Automaton. Since $f \neq \_$,[13] arc ${}_4(\_.0)_2$ will not be proposed, so only the arc ${}_4(0.0)_2$ remains a possibility. Note that because of the way the parser treats zeroes, the Ablaut Automaton, in essence, receives a free turn, so aside from proposing [–Subjunctive, –Plural] to the output, which already includes [+Preterite] from ${}_1(\text{a.e})_4$, it will also propose ${}_2(=.=)_3$ to the lexicon, attempting to consume the *f*. Since the string *gef-* is a

[10]Aside from its resemblance to Dolan (this volume)'s parser for Indonesian reduplication and circumfixation, this strategy owes a good deal to the Constraint Propagation strategy of Barton et al. (1987), Ch.6).

[11]For convenience we will adopt this notation, $_{\text{start-state}}(\text{label})_{\text{target-state}}$, when talking about arcs.

[12]There are nouns of Icelandic which begin *ga-*, but the inclusion of lexical features like [+V,–N] would take care of this ambiguity (thus barring the appearance of verbal inflection on nouns). Features like these would also be required in order to extend the parser's coverage to derivational morphology.

[13]The underline symbol, explained below, here corresponds to the length-mark ':'.

coherent lexical path this arc will remain live, and the automaton will jump to state 3 (the designated final state).

Now the Affixation Automaton will begin to scan for agreement suffixes. It will thus propose the arcs $_1(s.0)_2$ and $_1(0.0)_6$ and their associated syntactic feature sets [–Subjunctive, +Preterite, –Plural, +You] and [–Subjunctive, –Plural]. The input to the Ablaut Automaton is a zero in both cases, which is the only acceptable input, since there are no arcs out of its state 3, and so the feature set, since it is not checked on any proposable arc of the automaton, is passed straight through to the output by convention. However, the underlying morphosyntactic representation of *gafst* is no longer empty, so the sets output by Affixation, must be checked against the set [–Subjunctive, +Preterite, –Plural] already output by Ablaut's arcs $_1(a.e)_4$ and $_4(0.0)_2$. The first set ([–Subjunctive, +Preterite, –Plural, +You]) matches, correctly, and the feature [+You], which matches an as yet unspecified value in the morphosyntactic representation, is added to that representation.

A problem arises in that the second set ([–Subjunctive, –Plural]) also matches, and so does its vacuous phonological label. In this case no permanent problem arises, since the transducer must still deal with the remaining string *-st*, which does not match the expectations set up by Affixation state 6 (it expects the zero ending of, e.g., the Pret.1Sg. *gaf*). However, this specific problem is an example of a much larger one, so we might seek a more general solution for times when the phonological form of the string does not clear things up.

The problem is that the zero ending which state 6 expects is the default singular ending for this paradigm, appearing in the Pres.1Sg. and the Pret.1&3Sg. Where rules are ordered disjunctively, we expect the Elsewhere Condition to apply, which requires that where two or more rules match the input the one with the most specific Structural Description will win out. Looked at another way, we expect the default rule to operate only where some more specific rule is not available.

For the parser, two possibilities suggest themselves for implementing the Elsewhere Condition. The simplest is to do nothing, i.e. to allow both proposals to persist, causing the Affixation Automaton in this case to activate both state 2 and state 6. Since the features added to the morphosyntactic representation by $_1(0.0)_6$ are a subset of those added by $_1(s.0)_2$, they will simply be "lost in the sauce," as it were. If this strategy, by allowing a state to be activated erroneously, leads to intolerable ambiguity, enough information is available to the parser to prevent arc $_1(0.0)_6$ from ever being considered. Thus we can implement the Elsewhere Condition directly as a condition on the proposal set of a rule, immediately killing off any proposal which differs from another proposal only in specificity. For example, given the proposals "output a 0, and [–Sb, –Pl]" and "output a 0, and [–Sb, –Pl, +Pr, +2]," the first can be automatically discarded after a simple unification test.

4.4 Representations

The phonological representations we have seen so far have been simple strings of atomic segments, but it is uncontroversial that the structures on which listeners' sound-systems actually operate are not like strings, but rather much richer structures at very least like sets of strings interconnected in complex ways at all points. Since the family of automata of which our transducers are members requires that input be scanned one step at a time, and that a move of the automaton be rigidly defined, such complex representations present a severe difficulty. Automata are traditionally seen as scanning strings of symbols written on an abstract tape, one symbol at a time. Discovering an alphabet of symbols in which "two (or more)-dimensional" autosegmental representations can be written on a "one-dimensional" tape will occupy us in what follows.[14]

Let us refer to the alphabet of symbols which compose the input to an automaton as Σ. Given an autosegmental representation, it will be our task to derive a unique string of Σ-symbols, from which that representation is unambiguously recoverable. The immediate problem is not in representing the autosegmental tiers—for the time being we can consider these as strings of atomic symbols like '+' and '–', or 'C' and 'V.' The problem is in representing associations between tiers without the help of graphic association lines. As a first step in the development of Σ, we will rewrite our autosegmental representations on a tape which has been partitioned horizontally into tracks, corresponding to autosegmental tiers. Each track is scanned separately, and makes an autonomous contribution to the current state of the automaton. In order to represent associations on these multi-track tapes, we will adopt several conventions from musical notation, playing on the obvious similarities. In standard autosegmental theory (cf. Clements and Keyser (1983)), the timing of autosegmental effects is held to be coordinated by an almost featureless CV-Tier (or an even simpler X-Tier) whose function is simply to count the beat, as it were. Here, marks on the beat-track will be held to mark periods during which associated phenomena on other tracks occur. Thus placement on the tape is held to mark relative timing; association lines are implicit in this placement and are unambiguously recoverable from it. We give some formality to these notions in the following conventions.

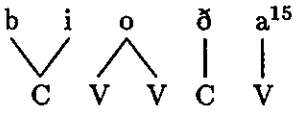
(7) Alignment Convention
Autosegmental specifications whose onsets are simultaneous are aligned vertically on the tape.

[14] On an alternative view, we might consider ourselves to be attempting the definition of a transducer's "field of vision," the region of an autosegmental representation which it can "see" at one time.

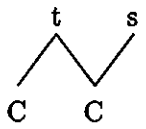
(8) Spreading Convention
An autosegmental specification persists until another specification on the same track succeeds it.

Thus we might translate the representations in (9) into tapes like those in (10).

(9) (a) *bjóða* 'to offer'



(b) Italian *zz*



(10) (a) b i o ð a
C V V C V

(b) ... t s ...
... C C ...

We can now slice the tape into vertical samples in accordance with the Symbol Allocation Convention given in (11).

[15]Depending on how one considers the phonetic spell-out of representations like these to work, this representation, traditionally transcribed with two phonetic segments [*bj*...], may appear to define a palatalized consonant [*b'*...], or a strange sort of contour segment [$\widehat{bj}$...]. Note, however, evidence from LuGanda (Clements (1986)) (cf. also Poser (1986) on Japanese) wherein glide-formation rules are written in this way. The reason for this is to account for the compensatory lengthening of the following vowel which results. Icelandic presents evidence that the underlying form of this stem is */beuð-/*: ablauted forms include *bauð* (a-Ablaut) and *buðum* (zero-Ablaut) (cf. *bresta, brast, brustum*, from zero grade *brst-*). Thus there is evidence for compensatory lengthening and the representation above seems justified. However, if this analysis fails to convince, one might adopt the representations

b i o ð a, b i o ð a.
| | | | | C V V C V
C V V C V

This would seem to make the prediction that the long vowel derived by compensatory lengthening/glide-formation would surface somewhat shorter than other long vowels. I do not know what the phonetic evidence is in this case.

(11) Symbol Allocation Convention
Symbols of Σ are allocated according to the rate of change on the most rapidly varying tier.

Alternatively,

Fluctuation on any tier causes a new symbol of Σ to be allocated.

Thus we can turn the tapes in (10) into the strings of ordered pairs in (12), where an underline corresponds to blank tape, and marks the perseveration of the preceding specification.[16]

(12) (a) $< b,C >< i,\_ >< o,V >< \_,V >< \eth,C >< a,V >$

(b) $\ldots < t,C >< \_,C >< s,\_ > \ldots$

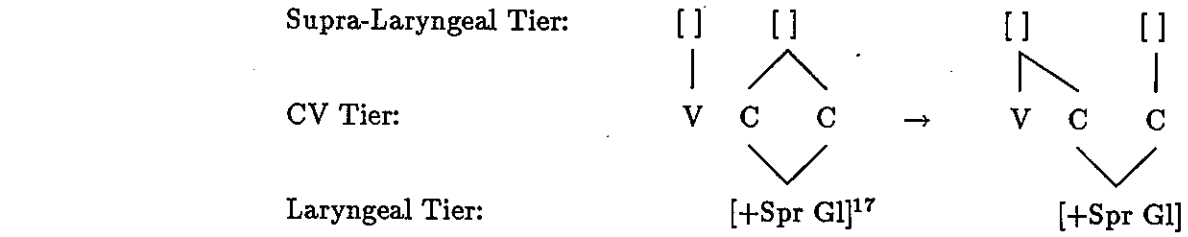
Up to now our phonological representations have been the maximally simple realization of the basic tenet that melodic and prosodic components of a phonological form ought to be given separate representations. We consider in what follows how to expand the basic notions we have developed so far to capture advances in the representation of melody (specifically the tree-structured hierarchy of tiers) and of prosody (X-Theory and Moraic Theory).

Recent work (Clements (1985), Walli-Sagey (1986), Steriade (1987)) has attempted to demonstrate that assimilation rules, seen as autosegmental spreading on some subset of tiers, seem to select out only some of the logically possible subsets of features to spread. This has suggested that features should be organized into a tree-structured hierarchy, with the additional condition that only well-defined constituents of this tree can spread. With this device a more constrained theory of assimilation can be developed which correctly fails to capture unobserved phenomena.

Up to now it has been tacitly assumed that Σ-symbols were ordered pairs of the form <melodic unit, prosodic unit>. If we further assume that the melodic unit (call it M) is a set of features $[f_1,f_2,\ldots,f_n]$ (corresponding to tape tracks $t_1,t_2,\ldots,t_n$), then we can capture the notion of a tree-structured hierarchy of features by assigning to M a labelled bracketing corresponding to whichever version of a feature-tree one wishes to implement. Consider as an example a version of the Icelandic rule of Preaspiration, which turns the first half of voiceless geminate stops into an h, onto which the supra-laryngeal features of the preceding vowel spread.

[16] It is suggestive to note the similarity, called to my attention by Bruce Hayes, between these representations and the "phonetic segments" of spectrogram readers and some speech recognition theorists.

(13) Preaspiration (after Thráinsson (1978))



e.g., /hatt-/ → [haht-] 'hat'

If we consider, along with the authors cited above, that Supra-Laryngeal and Laryngeal features are dominated by two daughters of the top-level node (called the root), and filling in the remaining hierarchical detail, we might write the relevant sequence *aht* from [*haht-*] as in (14).

(14) Representation of *aht*

$$a = < [[_{SupLa}[_{Pl/M}[_{Man}\ -cons][_{Pla}[_{Dors}\begin{bmatrix} +back \\ +low \end{bmatrix}]]]]], V >$$

$$h = < [[_{SupLa}\text{-}][_{Lar}\ +SprGl]], C >$$

$$t = < [[_{SupLa}[_{Pl/M}[_{Man}\begin{bmatrix} -cons \\ -son \\ -cont \end{bmatrix}][_{Pla}[_{Cor}\begin{bmatrix} -ant \\ -dist \\ -stri \end{bmatrix}]][_{Lar}\text{-}]], C >$$

(SupLa=Supra-Laryngeal, Pl/M=Place-Manner, Man=Manner, Pla=Place, Dors=Dorsal, Cor=Coronal, Lar=Laryngeal)

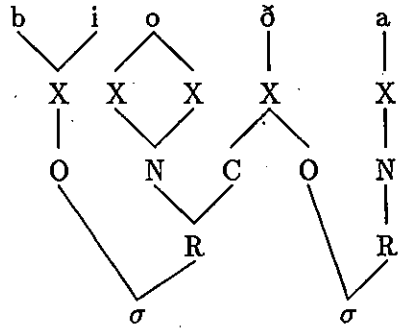
The important point is that in this formalism a melodic constituent can inherit its specification from a preceding constituent even when it is part of an otherwise distinct structure. Thus this formalism, even when enriched to include tree structure, can still represent "improper" bracketings.[18]

[17] The relevant distinction among Icelandic consonants is actually not voiced/voiceless but unaspirated/aspirated, hence the use of the feature [Spread Glottis]. The use of 't' instead of 'th' is an orthographic convenience.

[18] This approach is very reminiscent of the use of indices (B. Hayes, class lectures) to mark associations. The only real difference is that, given the Alignment Convention, it is not necessary to give subscripts names (or to keep track of those names). This impoverishment of the representation also places very strict requirements on the interpretation of the perseveration character: it must be

Considering now how to expand the prosodic element of a Σ-symbol, we note that Lowenstamm and Kaye (1986) and Levin (1985) have proposed that prosodic terminals are utterly featureless, the syllabic/non-syllabic distinction being properly captured in assigning the prosodic terminals a syllabic constituency. Thus, our familiar example might be represented as in (15), with 'X' marking the prosodic terminal (and assuming that the *ð* of *bjóða* is ambi-syllabic to make an illustrative point).

(15) (a)



(b) <b,[X,O]><i,[_,_]><o,[X,N]><_,[X,_]><ð,[X,C]><_,[_,O]><a,[X,N]>

More recently, a much more impoverished theory of prosodic constituency has been advanced in McCarthy and Prince (1987) and Hyman (1985). According to this Moraic Theory, the constituents of a syllable are moras, and melodic trees may attach directly to them. This theory accounts nicely for the range of prosodic templates appearing in templatic morphologies and reduplicative morphologies, as well as capturing neatly facts of syllabic weight.

The Moraic Theory also dispenses with the need for any prosodic constituents below the mora. Moras are attached directly to melodic root nodes. To see how we might include this in our Σ-notation, consider the Icelandic Re-Timing Rule, which takes underlying vowels and diphthongs and makes them long in open syllables and short in closed syllables. Thus to represent the four possible cases of long and short vowels and diphthongs which arise from Re-Timing, we might do as in (16)–(19).

associated with the preceding specification on the same tier. Thus geminates and partial geminates cannot be broken up by epenthesis or the perseveration character loses any possibility of well-formed interpretation. In this way we capture the phenomenon of Geminate Integrity (cf. Hayes (1986), Schein and Steriade (1986)) as a sort of locality condition on co-indexation.

(16) Long Vowel: *hatur* [ha:tʏr] 'hate'

(a)
h	a		t	ʏ	r
μ		μ	μ		
σ			σ		

(b) $< h,[\mu,\sigma] >< a,[\_,\_] >< \_,[\mu,\_] >< t,[\mu,\sigma] ><\text{Y},[\_,\_] >< r,[\_,\_] >$

(17) Short Vowel: *hattur* [hahtʏr] 'hat'

(a)
h	a	h	t	ʏ	r
μ		μ	μ		
σ			σ		

(b) $< h,[\mu,\sigma] >< a,[\_,\_] >< h,[\mu,\_] >< t,[\mu,\sigma] ><\text{Y},[\_,\_] >< r,[\_,\_] >$

(18) Long Diphthong: *heitur* [hei:tʏr] 'hot,' m.

(a)
h	e	i	t	ʏ	r
μ		μ	μ		
σ			σ		

(b) $< h,[\mu,\sigma] >< e,[\_,\_] >< i,[\mu,\_] >< t,[\mu,\sigma] ><\text{Y},[\_,\_] >< r,[\_,\_] >$

(19) Short Diphthong: *heitt* [heiht] 'hot,' n.

(a)
h	e	i	h	t
μ			μ	
σ				

(b) $< h,[\mu,\sigma] >< e,[\_,\_] >< i,[\_,\_] >< h,[\mu,\_] >< t,[\_,\_] >$

Thus we can see that the device of writing autosegmental representations on multi-track tapes, and deriving strings of symbols from slices of these tapes, allows us to bring virtually all of current generative phonology and morphology within the scope of automata theoretic computational models.

4.5 Conclusion

While this account of the operational underpinnings of IceParse has been necessarily sketchy in places, and certainly incomplete as an account of Icelandic morphology and phonology, I hope that the architecture of the system is clear enough. The focus of this model, it should be borne in mind, is to present a concrete role for

the competence grammar in processing. The parser is presented with a decision space of possible analyses which it must traverse in order to locate the correct one. The more constrained its path in the face of ambiguity the more efficiently it can compute its output representations. Given the first-pass assumption that the input is a string of the language, the grammar which generates strings of the language is a reasonable source of constraint. Modeled as a derivation in reverse, and executed by a pseudo-parallel non-backtracking transducer cascade, the grammar can be used for just this purpose, while exhibiting some of the real-time behavior that humans display.

The design (and implementation) of a model is by no means the end of a modeling project. The usefulness of models is that, given the inclusion of psychological constraints, they resemble experimental subjects, but ones which are transparent and plastic. Thus, given a model such as IceParse, we might expect to be able to perform several kinds of experiments, which fall in general into two classes.

First, one can perform experiments in "impaired processing," by which I mean in particular the processing of ungrammatical, distorted or noisy input. The problems which arise are that either too many arcs will light up, overwhelming the parser's ability to discriminate, or that no arc will light up, blocking the parser's progress. In the first case, we might experiment with combining transducers into larger clusters, thereby increasing the number of states and the specificity of transitions. In the second case, we might experiment with ways to determine the "closest match," or some set of adequately close matches, in order to force a transition and continue the parse. Such experiments ought to generate predictions as to which analysis/analyses people will assign to bad input.

The second class of experiments involve something more like normal processing, and revolve around the notion of underspecification of the input. This class of experiments would seek to determine, essentially, how many features the parser can do without. The purpose of such experiments would be to generate predictions relevant to Speech Recognition work as to which features are most important to key on in building models of low-level acoustic signal processing, and which features, in the best case, can safely be ignored.

Chapter 5

A Syllable-Based Parallel Processing Model For Parsing Indonesian Morphology

William B. Dolan

5.1 Introduction

The role of a morphological parser, ultimately, is that of a component of a larger natural language processing system, providing the syntactic and semantic modules with the information they require for the larger tasks of sentence and discourse processing. Depending on the language under consideration, such information might be as simple as the word's syntactic category and a gloss. However, languages with more complex morphosyntactic relationships will require that the parser provide the syntax with much more information about the word, including facts about agreement, case, and any other syntactic features which are marked morphologically.

In any case, it is clear that the sort of information required by higher level parser components can be very different from that which must be considered internally by the morphological module. For example, while facts about the presence, ordering, and shapes of particular formatives may prove crucial to correctly parsing a given word, this sort of information is useless to the syntax and semantics. From a syntactic standpoint, the difference between a nominal produced by reduplication and a nominal produced by circumfixation is irrelevant; the important fact is that each functions as a noun.

The morphological parsing component's task can vary dramatically from language to language. In the extreme case, parsing may be unnecessary, with morphological processing reducible to simple lexical lookup. For classically agglutinating languages, the task still seems relatively simple: 'undo' any phonological rules which might have applied during the course of the word's derivation, and output the list of component formatives and their glosses. Ideally, an overall gloss and syntactic category can be arrived at by combining these formatives according to some regular formula (perhaps as simple as straightforward addition). Turkish, with its highly agglutinating character and minimum of morphophonemic interactions, is quite amenable to such a parsing strategy (Hankamer (1986)).

However, this paradigm is of little value when applied to a language with nonconcatenative morphology; an interlinear morphological gloss is an impossibility for words in Arabic or Hebrew, for example, since phonological material representing two separate morphemes may be interwoven in the temporal segmental sequence. Similarly, languages with highly synthetic morphological processes or complex morphophonological interactions pose problems for a model which assumes that parsing is a left to right affix-stripping procedure, since the underlying phonological shape of a formative may be very different from its phonetic form.

On the surface, at least, Indonesian morphology seems well-suited to a parser which simply provides interlinear glosses for input words. The system is basically agglutinating, with a relatively small inventory of phonological and morphophonological rules producing surface phonetic forms. Boundaries between formatives are relatively clear, making a straightforward affix-stripping procedure an attractive parsing strategy. However, despite its similarities to agglutinating systems like Turkish, Indonesian morphology does not readily lend itself to the sort of left to right parsing model sketched above.

Instead, I will argue that correctly assigning a meaning to an Indonesian word's sequence of formatives depends crucially on first identifying the word's root and the morphological structure into which the formatives are arranged. Since these pieces of information are not necessarily available at the left edge of a word, this claim presents a paradox for simple left to right parsing models: parsing begins at the left, but a correct parse may be dependent from the outset on information which will become available only when the right edge of the word is reached.

I will be assuming an approach to morphology similar to that in Aronoff (1976), with word-formation rules of the type $[X]_N \longrightarrow [X + \text{affix}]_V$ generating forms which do not exist as lexical entries. Rules specify the phonological, syntactic, and semantic effect of adding a particular formative to a base form.

This paper is arranged as follows: section 5.2 consists of a description of Indonesian word-formation processes, section 5.3 details some of the problems this

morphological system poses for a parser, and finally, section 5.4 represents a discussion of current morphological parsing models and a description and discussion of an alternative model for Indonesian. This model has been partially implemented in Pop-11 running on a Sun 2 workstation in the Linguistics Department at UCLA. (Coding details of this implementation have been left out in favor of a more general discussion of the algorithm's behavior.)

5.2 Review of Indonesian Morphology

The following description of Indonesian morphological facts is by no means exhaustive; more complete discussions are provided in Lapoliwa (1981), Macdonald and Darjowidjojo (1967), Verhaar (1984), and Wolff (1980). The focus here is on productive rather than lexicalized morphological processes, and on aspects of Indonesian morphology which make it especially problematic for current parsing models.

5.2.1 Productive Affixation and Reduplication

Practically all Indonesian morphology can be classified as derivational, although a few word-building processes might be argued to have inflectional properties. The inventory of productive affixes is not large; there are seven or eight prefixes and three or four suffixes, but the range of morphological structures these generate is augmented by the possibility of combining several of the prefixes and suffixes into circumfix or prefix-suffix patterns. Indigenous morphemes (as well as many borrowed ones) conform to a fairly strict set of phonotactic patterns. Prefixes and suffixes are all monosyllabic, while roots normally have two or three syllables. A list of the productive prefixes, suffixes, and circumfixes in modern Indonesian is presented in Appendix A.

In addition to affixes, Indonesian displays an interesting array of process morphemes, with at least three distinct types of reduplication: full, partial, and 'imitative.' Full reduplication copies the entire root or root plus affixes, with regular semantic results depending on what level morphological constituent is copied and the syntactic/semantic class of the unreduplicated form. For example:[1]

[1] In these examples, '+' indicates simple affixation, and '–' indicates reduplication.

(1)

a.	bunga	'flower'
	bunga–bunga	'all kinds of flowers'
b.	se+kali	'one time'
	se+kali–kali	'from time to time'
	se+kali–se+kali	'once in a while'

The other two process-formatives, partial and imitative reduplication, will be described in more detail in section 5.2.4; in any case neither is productive, but it is not entirely clear that these phenomena can be relegated to the lexicon (for a discussion of some of the formal properties of Indonesian reduplication see Cohn (1987)).

5.2.2 Circumfixation

Circumfixes are used extensively in Indonesian word formation. Circumfixation involves simultaneously affixing a prefix and suffix to a base form, with the resulting semantic effect underivable from the compositional meanings of the independently occurring affixes. For example, the prefix *kə* attaches to numerals to form ordinal adjectives (*tiga* 'three', *kə+tiga* 'third'), while the suffix *an* forms concrete nouns from verbal or nominal roots (*makan* 'eat', *makanan* 'food'). However, when both *kə* and *an* appear on a verb or adjective root, the resulting form is an abstract nominal, a semantic effect which seemingly cannot be derived by combining the meanings of the individual affixes:

(2)

besar	'big'	bangun	'arise'
kəbesaran	'bigness'	kəbangunan	'awakening'

Non-semantic arguments can also be constructed favoring the claim that these affix combinations really do represent a single discontinuous morpheme, rather than two separate formatives. Assuming that circumfixes are the result of the cyclic addition of two separate affixes fails to account for forms like *penarian* 'dancing', which must be treated as a combination of the verb root *tari* 'to dance' and the nominalizing circumfix *pəN+an*. In their independent affix-forms, both the prefix *pəN* and the suffix *an* act to nominalize verb roots. However, neither of the substructures [*pəN* [*tari*]] nor [[*tari*] *an*] can combine with a nominalizing affix like *pəN* or *an* to produce a well-formed word. In other words, claiming that a form like *pənarian* 'dancing' consists of three separate formatives would entail the addition of a nominalizing affix to what is already a derived nominal:

(3)

a. **penari** 'the dancer':

pəN+tari	[nominalizing prefix + 'to dance']
*[pəN+tari]+an	['the dancer' + nominalizing suffix]

b. **tarian** 'a dance':

tari+an	['to dance' + nominalizing suffix]
*pəN+[tari+an]	[nominalizing prefix + 'a dance']

c. **penarian** 'dancing':

pəN+tari+an	['to dance' + nominalizing circumfix]

Structures like *[[*pəN* [*tari*]] *an*] and *[*pəN* [[*tari*] *an*]] violate Indonesian affixation rules, since *pəN* cannot productively combine with nouns, and the only noun-type with which *an* can productively combine is the subclass of numbers. Since cyclic application of prefixation and suffixation rules cannot account for forms like *penarian*, a circumfixal description seems inescapable.

5.2.3 'Infixed' Prefixes

Both transitive and intransitive verb roots can be reduplicated to indicate intensity. These reduplicated roots normally co-occur with prefixes as in pattern (4a) below. Additionally, the verbal prefix *məN* participates in the morphological pattern shown in (4b):

(4)

a.	prefix+X–X	*mənulisnulis*	'write: intensive'
b.	X–prefix+X	*tulismənulis*	'write: reciprocal'

In (4a) the prefix *məN* is straightforwardly adjoined to a reduplicated transitive root (*tulis* 'write'), while in (4b), the affix becomes a quasi-infix, appearing between the halves of the reduplication. The two prefixation/reduplication patterns in (4a) and (4b) have quite distinct semantic effects, with the structure shown in (4b) consistently marking reciprocality of the verb's action. Structure (4a) would seem to be the semantically unmarked case; reduplicating a verb root typically yields this connotation of intensity or iterativity, and prefixing *məN* to a stem is the normal means of creating transitive verbs. Even in structure (4b), *məN* is unambiguously a prefix, following the same allomorphy rules regarding nasal assimilation/stop deletion that it would follow when in normal prefix position[2] (the specifics of this morphophone-

[2]In fact, the morphophonemic rule triggered by *məN* (discussed in section 5.2.6) "overapplies" in forms like (4b), producing *mənulisnulis* rather than **mənulistulis*. This might provide evidence for the cyclic application of phonological rules, with the morphophonemic *məN* rule acting before reduplication copies the base.

mic rule are discussed in section 5.2.6).

5.2.4 Non-productive morphology

There are two common infixes in Indonesian, but these seem to be of purely historical interest. Forms which are historically infixed are appear to be lexicalized:[3] no clear semantic effect can be assigned to these infixes, and speakers are not generally conscious of the infix as a distinct entity in words like the following:

(5)	gembung	'swollen'		getar	'tremble'
	gəlembung	'swollen'		gəmetar	'tremble'

Partial reduplication in Indonesian copies the first consonant of a root, prefixing *Cə* to the root. Thus:

(6)	laki	'male'	langit	'sky'
	ləlaki	'male, man, husband'	ləlangit	'ceiling'

Imitative reduplication involves the affixation of a root-like form to the root:

	balik	'reverse'	⟶	bolak–balik	'to and fro'
(7)	umbang	'float'	⟶	umbang–ambing	'drift to and fro'
	erot	'crooked'	⟶	erang–erot	'zig-zag'

These forms generally carry a connotation of randomness, variety, or repetition. Although there are several fairly regular vowel and consonant alternation patterns involved in this sort of reduplication, there are numerous exceptions; both partial and imitative reduplication appear to be lexicalized rather than productive morphological process in modern Indonesian. Partial reduplication was historically a means of forming plurals, but this is no longer the case, and no constant semantic link can be set up between simple and partially reduplicated forms.

Interestingly, although the three kinds of Indonesian reduplication differ tremendously in form and productivity, speakers treat forms that are reduplicated in any way as a distinct class in the language. Words produced via partial and imitative reduplication are subject to the same general constraint against re-reduplicating a form that blocks forms such as:

[3] In generalizing the design of the parser, it would clearly be necessary to include infixes as parse-worthy; the inflectional infixes of Tagalog, for example, could not simply be treated as part of the root.

(8) *kuda–kudaan—kuda–kudaan
cf. kuda–kudaan 'toy horse'
kuda 'horse'

Reduplication combined with *an* suffixation is a productive means of forming nouns which refer to 'toys', a construction which imparts no sense of plurality. However, the normal pluralization process for nominals, reduplication, is impossible for *kuda-kudaan*. At some level, speakers are capable of recognizing a form as reduplicated; whether this process should be treated as phonological or lexicalized is not entirely clear.

5.2.5 Morphological Combinations: Word Structures

A word in Indonesian can consist simply of a root, while morphologically complex forms generally have no more than two or three layers of affixes. The maximal non-compounded word consists of two prefixes, a root, and two suffixes (the last of which being one of the possessive enclitics *mu, ku, or ña*). Additionally, there may be a single reduplication at some level in the word; either monomorphemic or morphologically complex stems can be productively reduplicated (as described in section 5.2.1). Reduplicated bases of any type can be combined with affixes in most of the same patterns that unreduplicated forms can, provided that the general constraint against re-reduplicating a form is not violated.

5.2.6 Morphophonemic Interactions

Several Indonesian prefixes trigger phonological changes in the root, causing the initial root segment to either drop or mutate. For example, the final nasals in the verbal prefixes *məN* and *pəN* are generally treated as unspecified for place of articulation features. When one of these prefixes is followed by a root-initial voiceless stop or nasal, the underspecified nasal *N* assimilates in place of articulation to the following consonant, which acquires nasality from it. A general rule of degemination reduces the seqence to a single nasal consonant.

(9)	pinjam	(bound verb root)		tulis	(bound verb root)
	məminjam	'to borrow from'		mənulis	'to write'

If a vowel or voiced obstruent follows the prefix, *məN* and *pəN* surface with a velar nasal as [*məŋ*] and [*pəŋ*].

5.2.7 Other Considerations

Among the morphological processes which I have chosen not to discuss is compounding, which appears to be largely lexicalized (compound bases function like simple bases in terms of affixation and reduplication). Additionally, in the interests of expedience, certain productive phonological rules such as 'glide insertion' (see Lapoliwa (1981) for a description) have not been treated, and a number of marginally productive affixes have been treated as lexicalized.

5.3 Theoretical Issues in Parsing Indonesian

5.3.1 Phonology/Morphophonemics

Most immediate among the problems posed for a parsing model by Indonesian morphology are the effects of phonological and morphophonological rules which obscure the shape of formatives. Indonesian's *məN* rule (section 5.2.6), for example, has potentially disastrous effects for a parser, because it drastically alters the shape of both the affix and stem. How is lexical access to proceed if not even the first segment of the stem is intact in the surface form?

This sort of problem is hardly unique to Indonesian, however, and the morphological parsing literature describes various strategies for arriving at the underlying form when faced with a mismatch between underlying and surface forms. The relative merits of some of these techniques will be discussed in more detail in section 5.4.

5.3.2 Semantics

More problematic is the impossibility of stating a constant semantic/syntactic effect for many highly productive Indonesian formatives that will adequately describe their role in word-formation. A single formative's effect can vary dramatically depending on what type of root/stem it interacts with. For example, reduplicating[4] different types of simple roots yields very different semantic effects:

(10)

REDUP–noun	⟶	N:	'plural/variety'
REDUP–time nominal	⟶	Adv:	'when X is in effect'
REDUP–adj	⟶	Adj:	'intensive'
REDUP–verb	⟶	V:	'iterative; intensive'

[4]Reduplication has been treated here — somewhat arbitrarily — as a form of prefixation.

This provides a very clear example of why an interlinear gloss would prove useless for Indonesian morphology. Reduplication is a *process* which produces different effects when applied to different sorts of bases; it is not a 'morpheme' whose meaning can simply be added to the base's meaning to produce a new word. Furthermore, the particular effect of reduplicating a form is not dependent solely on the base's syntactic category, but may also depend on certain semantic characteristics of the base form. This is illustrated by the first two patterns in the set above: reduplicating a 'time-related nominal' such as 'day' or 'hour' produces very different effects from reduplicating other nominal bases. There are a number of these semantic subclasses to which many Indonesian morphological rules appear to be sensitive: for example, numbers, [+human] nominals, places, and mass vs. count nouns are all treated differently by individual rules, as are stative verbs, 'psych-verbs' such as 'love' or 'feel', and verbs whose meaning are connected to some sense of finality (i.e., 'death', 'loss', 'theft', 'finishing', etc.)

5.3.3 Semantically Irregular Forms

Productively 'Idiomatic' Formative Combinations

It thus appears that a morphological parser for Indonesian requires a great deal of information about a word's root before any definite role can be assigned to the formatives associated with it. In most cases, knowlege of normal rules for combining stems with derivational formatives, as well as information about the root's syntactic and semantic class, will allow parsing to proceed correctly. However, for a whole class of words, this information is insufficient. Certain productive formative combinations produce an 'idiomatic' effect which is underivable from meanings of the individual formatives and normal stem-formation rules:

(11) [REDUP–prefix+VT]
e.g. *tulis–mən+tulis* 'write: reciprocal'
(example (4b) from section 5.2.3)

In this case, although the formatives involved are identical to those in the word *menulisnulis* 'write: iterative', the structure into which the formatives is arranged produces a crucially different semantic effect. Another example is:

(12) [REDUP–Adj+an] 'Adj-ish'
e.g. *gelap–gelap+an* 'dark-ish', 'dusky'

According to normal affixation rules (Appendices B and C), this combination of formatives should produce an intensified adjective with a distributive sense (rules 7

and 37, Appendix C). Instead, the structure [[Adj–Adj]+*an*] produces a weakened form of the adjective, meaning 'Adj-ish'. There are at least five such morphological structures in Indonesian[5] structures in which formatives combine to produce semantic effects which are not predictable from the normal rules governing formative-combinations.

Lexicalized Morphologically Complex Words

A final class of problems is posed by Indonesian words such as:

(13)

laba	'profit'
laba–laba	'spider'

As described in section 5.2.3, nominal reduplication normally indicates plurality. Thus, *laba–laba* might be expected to mean 'profits', instead of its actual meaning, 'spider.' Such "idiomatic" Indonesian reduplications are analagous to English words like *reply* and *shiftless*: while their surface forms might suggest that these are regularly-derived polymorphemic words meaning 'ply again' and 'without shift', their meanings are in fact not predictable. The parsing procedure which might be assumed to assign proper meanings to words like *happiness* (*happy* + nominalizer) or *redo* (*again* + *do*) must be blocked in such instances. Wehrli (1985) has argued that such words represent an argument against a morpheme-based lexicon, since a parser with knowledge of morphological structure/rules would mistakenly analyze such words as:

(14)

re	+	ply	REDUP	–	laba	
'again'	+	'fold'	plural	+	'profit'	

In fact, such forms are never ambiguous; a parser must provide only the lexicalized parse as its output. Similarly, Indonesian displays instances of lexical items which appear to be reduplicated but which are nonetheless monomorphemic. These words do not have unreduplicated counterparts:

(15)

gigi	'tooth'	cf. *gi
dada	'chest'	cf. *da

Interestingly, these words obey the constraint against re-reduplication discussed in 5.3.3. In some way, then, speakers think of them as reduplicated, and the forms in (16) do not occur.

[5]These five structures are presented as the first set of disjunctively ordered rules in Appendix B, labeled 'Idiomatic Structures Block'.

(16) *gigigigi
*dadadada

5.3.4 Information Required Internally by the Parser

Thus, in order to be useful to the syntactic and semantic modules of a natural language processor, a morphological parser for Indonesian must take into account at least the following information about a word:

- What formatives are present, including both affixes and reduplication.
- The morphological pattern into which the formatives are arranged, at least in those cases where this arrangement has a semantic effect. This includes facts about whether a circumfix occurs, what level of morphological constituent is copied by any reduplication, whether the formatives are arranged into one of the 'idiomatic' word structures described in 5.2.3 and 5.3.3, and so on.
- The syntactic class of the stem to which the derivational/inflectional formative attaches. For some stem classes, semantic information also proves crucial: for example, 'time-related' nominals — roots like 'day', 'hour', etc. — behave differently from other nominals when combined with various formatives.

Additionally, some heuristic mechanism must be provided which will prevent the normal parsing process from acting on forms such as *labalaba* and *gigi*.

5.4 Parsing

5.4.1 General Considerations in Parser Design

The strategy of any morphological parser involves trying to map a phonetic form onto a unique formative structure, but very different approaches may underlie this common aim. A number of decisions concerning the parser's basic structure must first be made, taking into consideration such factors as the structure of the grammar being modeled, psycholinguistic plausibility, and computational tractability and efficiency. Below is a brief description of some of the major design features.

Top-down vs. Bottom-up

Given a set of phrase structure rules of the form 'S $\longrightarrow$ NP VP', a top-down (or 'goal-directed') parser attempts to satisfy the left side of each member of the set of rules, searching the input for constituents which correspond to those on the right

side of a given rule. A bottom-up (or 'data-driven') parser, on the other hand, works from the input data, attempting to build up combinations of constituents which will match the left side of one of the phrase structure rules.

Depth-First vs. Breadth-First

'Depth-first' vs. 'breadth-first' approaches to parsing tasks represent very different basic approaches to the task of deciding on a word or sentence's structure. The depth-first approach to parsing is illustrated by the 'keCi' model (Hankamer (1986)) for Turkish morphology, which pursues a single possible candidate parse to its conclusion, whether successful or failed, before checking any other possible parses for the input word. Using the depth-first approach, hypotheses about word-structure are tested serially, with the parser trying to identify a successful parse, then backtracking to check other possible hypotheses which might be consistent with the input data.

In contrast to this full, serial exploration of each candidate is the breadth-first approach, in which a number of different parses are pursued 'simultaneously,' advancing as a broad front through the set of phonological and morphological rules which make up the grammar. Such an approach relies on first building up a set of potential candidate parses, and then passing this 'cohort' of possible parses through the rules of the grammar, with gradually more and more of its members being ruled out as ungrammatical. After completing this circuit through the rules of the grammar, the only remaining candidate(s) should correspond to the correct parse(s) for the input word or sentence.

5.4.2 Modeling Human Perception

Each design choice has a crucial influence on the eventual form of the parser, and, as already noted, on its pretensions to any degree of psychological reality. From a purely results-oriented viewpoint, there is little reason to prefer any particular combination of values for the parameters described above: essentially the same grammar will be required no matter what sort of design is chosen, and the set of successful parse outputs must be the same. There may be substantial differences in factors such as computational tractability and efficiency depending on exactly what design is chosen, but I will ignore such issues; arguments based on ease of programming and processing times are more dependent on technical concerns than on any linguistically interesting facts.

Psycholinguistic considerations cannot be dismissed so easily, and should play a central role in the organization of any linguistically interesting parser. Particular

design choices suggest very different pictures of how human perception functions, and some designs make claims about the nature of perception which are clearly incorrect. For example, any parsing design which requires right-to-left processing of the input is dubious in light of experimental evidence for left to right effects in auditory perception (Emmorey (1986)).[6]

In other cases, the implicit claims made by a given model about how human perception works are more controversial: for instance, modeling linguistic processing as a depth-first rather than breadth-first behavior suggests that potential ambiguity in the input is not identified until after the first successful parse has been identified, for only then will backtracking force the parser to attempt a second parse. A breadth-first approach, on the other hand, assumes that knowledge of potential ambiguity is present at the outset of the parsing process, and that one of the functions of a parser is to decide among these competing hypotheses. This latter model is supported by experimental evidence: semantically ambiguous words appear to briefly prime all the possible readings for that word, suggesting that all readings must be recovered at some point in the lexical identification process (Emmorey (this volume)).

Parsing routines which operate in a breadth-first manner are quite compatible with the suggestion that human perception involves forming and testing a number of hypotheses about the data simultaneously, with various co-processors working in parallel to arrive at the correct solution(s). This view is perhaps epitomized by Parallel Distributed Processing (Rumelhart and McClelland (1986)), which models cognitive processes by positing simple, interlinked neural processors which act in parallel to simultaneously consider a number of solutions to a given problem. Such parallelism is claimed to have substantial explanatory advantages over models which posit serial processing for tasks like word perception.

In addition to these general psycholinguistic considerations concerning what the general design of the parser should be, there are also — potentially, at least — language-specific considerations which might affect the design. Models which rely on

[6]Recognizing left to right processing effects does not necessarily entail the assumption that the morphological parsing process itself proceeds left to right. It is possible that such effects might be explained in terms of a two-level model of the perceptual process, with the first level constituting a phonological buffer-filling mechanism which passes on to the parsing device as much of the input as is available at any given point. The parsing process itself might treat this input in a left to right manner, or right to left, or holistically. In such a system, left to right processing effects could be attributed to the accidentally left to right nature of the acoustic signal, rather than any inherent asymmetry in the parsing mechanism.

It should also be noted that opinions in the literature which concern the psychological validity of computational parsing models are often based on the performance of serial implementations of these models. This obscures the fact that strategies which are temporally unrealistic in a serial implementation might well prove feasible if implemented in a truly parallel fashion.

highly language-specific structures seem undesirable from a linguistic point of view, and the design of a universal morphological parser is a goal well worth pursuing. Yet even assuming that part of the language acquisition device consists of such a general morphological parser, there may be a set of perceptual parameters which are used differently by speakers of different languages in the parsing task. Indeed, in highly isolating languages (Mandarin, for example) the lack of complex morphology might obviate the need for any online morphological component. Wehrli (1985) claims that a static, dictionary-based view of the lexicon has strong advantages over a dynamic morphological parsing component for languages like French and English.

On the other hand, the complexity of morphological systems like those found in many American Indian languages make some sort of online parsing mechanism seem essential to linguistic processing. Although it is impossible to suggest arbitrary limits on the potential size of the mental lexicon, it is highly unlikely that speakers of a language with morphological systems approaching syntactic-level complexity (Creek or Eskimo, for example) depend on lexical lookup rather than some sort of morphological parsing strategy. Consider Finnish, which has a potential 2,000 inflected forms for each noun and 12,000 for each verb (Gazdar (1985)); this presents a compelling argument against positing simple lexical lookup as the sole morphological processing mechanism available to humans.[7]

The diversity of morphological systems across languages suggests that speakers of different languages might make use of different sorts of information as heuristics in expediting the parsing process. After all, languages display wildly differing degrees of complexity in their morphological, stress, and phonotactic systems; it would seem rather profligate if listeners did not use their knowledge of possible morphological and phonological word-structures to limit the search space involved in parsing a given input word. In designing this morphological parser, I have depended quite heavily on idiosyncratic facts about Indonesian word structure. To a certain extent this limits the generalizability of this parsing model, but at the same time it allows the parser to extract a maximum amount of information from the input with a minumum of false steps. Issues of generalizability will be discussed in section 5.4.8, but first I will discuss the parsing model and its advantages over other approaches for Indonesian morphology.

[7]In fact, lexical lookup and a view of morphological parsing as a dynamic process are not the only possibilities. Another is that morphological relationships are represented as a static set of relations among lexical items, with lexicon-internal mappings.

5.4.3 A Model For Indonesian

Perhaps the most influential of current morphological parsing models is the 'two-level' model of Koskenniemi (1983a), (1984). This paradigm, implemented for Finnish, models a language's phonology as a set of finite-state transducers, each representing a phonological or morphophonological rule. These transducers operate in a parallel array, directly relating surface/lexical segment pairs. (Hence the term 'two-level'; intermediate stages in the derivation do not exist in this model, contrary to generative models of phonology.) Transducers do not 'rewrite' or modify either the surface or lexical form, but simply function as grammaticality filters, with morphophonemic interactions and formative ordering constraints expressed in each formative's lexical entry. The entry for a Finnish noun stem, for example, includes information about which inflectional endings might follow it. Additionally, each entry contains a reference to the phonological transducer which will allow any phonological alternations conditioned by the ending to pass successfully through the phonology.

Koskenniemi notes that his system would require 'extensions or revisions ... for an adequate descriptions of languages possessing extensive infixation or reduplication' (Koskenniemi (1983a), p. 27). Gazdar (1985) is less optimistic, pointing out that the two-level model 'has no straightforward way of talking about tree-structured objects' (p. 6); rules are restricted to one-segment variations, while alternations which affect more than one segment must be lexicalized as suppletive. This approach is clearly inadequate for representing such common word-level dependencies as stress, circumfixation, and reduplication.

Such non-segmental dependencies are precisely the phenomena which the Indonesian parsing model described below is intended to treat elegantly. Circumfixation, reduplication, and 'idiomatic' word structures (section 5.3.3) all seem to require that the parser have access to the entire word, rather than some n–length segmental window. The approach adopted here provides such access; morphological parsing in Indonesian is assumed to be an essentially symmetrical procedure, with the parser free to range over all the formatives in a word in deciding on exactly what morphological dependencies exist.

To reiterate the major points from the description of Indonesian morphology presented in section 5.2:

- The set of productive affixes is quite small.
- There are few phonological/morphophonological rules.
- Affixing morphology is monosyllabic.

- Morphological structure — not just the meanings of a word's component formatives — is crucial.

- The same formative can have very different semantic effects depending on the value of the stem.

The last three points suggested the basic format of the parser, which assumes that identifying the word's root and its syntactic/semantic properties is an essential first step in processing a word in Indonesian. The regular monosyllabic character of affixes in the language provides the basis for an affix-stripping routine which divides the word into potential affixes and a potential root. Before the semantic effects of any affixes can be determined — in effect, before any real parsing can take place — the position and identity of the root must be discovered. This is in keeping with Cutler et al. (1985), who propose that stems take processing priority over affixes, and that the cross-linguistic preference for suffixing rather than prefixing morphology results from a tendency to place the perceptually more important element in the more salient initial position.

Segmentation Units

A fundamental problem for speech perception models lies in identifying the basic segmentation unit used by speakers in processing the incoming acoustic signal. Psycholinguists have debated this question for some years, with two major contenders emerging, the segment/phoneme and the syllable (Barry (1984)). The choice between these two remains unclear: there is considerable evidence supporting both hypotheses, and some psycholinguists have suggested that speakers of different languages may choose one segmentation strategy over the other based on concerns of processing efficiency. For example, Cutler et al. (1986) suggest that because of its highly irregular syllable structure and unclear syllable boundaries, speakers of English depend on segments as the basic acoustic segmentation unit, while French speakers rely on the syllable. They conclude that segmentation strategies in continuous speech perception may be language-specific, and that the basic segmentation unit may be a parameter set during the language acquisition process in order to take maximum advantage of the acoustic cues provided by that language.

5.4.4 The Parser

Indonesian's regular syllable structure, coupled with the fact that all of its affixes are monosyllabic, makes it an extremely attractive candidate for syllable-based pars-

ing. The identification of potential morpheme boundaries can be made much more efficient if they are constrained to coincide with syllable boundaries.

This is the tactic used by the phonological 'front end' of the Indonesian parser described here. An affix-stripping routine checks each syllable[8] in the input word against the small set of phonological affix-shapes and their allomorphs. All syllables must be treated as potentially part of a root; additionally, syllables which match some member of the affix-shape list are possible affixes. This bottom-up routine moves left to right across the input word, assembling a list of all possible underlying formative combinations for that sequence of input syllables. This stage can be viewed as a sort of syllabic 'triage' which allows the parser access to information about what the word's root and overall morphological structure might be.

Once the set of candidate formative-assignments has been assembled, the second stage of the parser becomes active. Each candidate parse is subjected to root lookup, with a lexical search being conducted for its putative root. If the root exists, then the candidate continues on to the next stage; if no such root exists the candidate is eliminated from further consideration. In the final parsing step, a set of Aronoff-style rules is applied to each surviving candidate parse, attempting to combine all of its formatives into a well-formed morphological complex. A candidate is unsuccessful when no rule exists to combine its stem with whatever unassociated formatives remain. What follows are more detailed descriptions of each of the parser's components: the phonological 'preparse', the root search, and the morphological matching stage.

Phonological Preparse

Input syllables are passed to the candidate generation routine in triplets. This restricts the operation of morphophonological and phonological rules to syllables which are immediately adjacent to the syllable currently under consideration. These rules are thus capable of resyllabifying part of the current syllable to the preceding syllable, altering the shape of the following syllable (as required by section 5.2.6's *mə N* rule, for example), and so on.[9] Each time potential change is encountered, the

[8] Although Indonesian's highly regular CV(C) syllable structure would make writing a syllabification algorithm a fairly trivial task, the current implementation of the parser assumes pre-syllabified forms as input.

Similarly, word divisions have been arbitrarily presupposed. If the parser were to be extended to deal with connected speech, then stress and timing information — which speakers clearly have access to — might be incorporated into word-segmentation heuristics.

[9] In a truly parallel implementation of the parser, all syllables in the input might be considered simultaneously, freeing the system from the arbitrary restriction that morphophonological and phonological rules have effects localized to adjacent syllables. However, whether or not this freedom

candidate-generation routine produces two candidates: one which assumes identity between the phonetic and underlying forms, the other reflecting the 'undoing' of the rule.

Take, for example, the input word [*kə ti ga*], 'third'. The first syllable will match one of the affix shapes, *kə*. This match generates two separate candidate parses for the first syllable of the input, both [*kə* prefix] and [*kə* root]. (The additional 'root' candidate is necessary because any syllable which matches one of the affix-shapes might also be a root syllable. Although indigenous Indonesian formatives are generally identifiable as roots or affixes based on solely phonotactic grounds, this is not a reliable test for syllable-classification, since borrowings into the language frequently violate these morpho-phonotactic constraints.) Any syllable which does not match one of the phonological affix-shapes is automatically classified as belonging to the root.

The 'undoing' of morphophonological rules is triggered on a purely phonetic basis: for example, the input syllables [*mə nu*] trigger a reversal of section 5.2.6's '*məN* rule'. The first syllable of the input, [*mə*], matches a member of the list of *məN* allomorphs. This triggers a routine which checks to see whether the next syllable begins with a nasal, as it must if [*mə nu*] represents the phonetic reflex of [*məN* + root]. In this case, finding [*n*] as the onset of the next syllable satisfies this requirement, causing the creation of a candidate in which a voiceless stop (homorganic with the phonetic nasal) is recreated as the underlying onset for the root following *məN*. A third possibility, that the surface [*mə nu*] represents the assimilated and degeminated form of an underlying [*N+n*], produces yet another candidate. The sequence [*mə nu*] could thus represent underlying:

(17) a. [*mə nu* root]
b. [*məN* prefix] + [*tu* root]
c. [*məN* prefix] + [*nu* root]

All three possibilities are generated and passed on to the morphological matching routines as candidate parses. The program iterates over the word, checking three syllables at a time, and adding to the cohort of possible underlying structures with each iteration.

is desirable from a phonological standpoint is another matter; although some classes of phonological rules (i.e. stress, vowel harmony, etc.) must assume that the whole word is present when rules apply, the effects of other kinds of rules must clearly be restricted to local environments.

Reduplication Check

Reduplication is also checked for at this 'phonological preparse' stage. Two different routines perform this function, based on how many separate sequences of consecutive root syllables exist in the candidate. If only one root sequence is found, then the only possible reduplication for that form will be root reduplication. Root reduplication is checked by a routine which searches across a lone root for a recurrence of its first syllable. If a copy of that first syllable is found somewhere in the root, then the root is divided into two parts, one consisting of the syllables preceding the second instance of the initial syllable, and the other consisting of the syllables following it. These parts are checked for identity; if they match, two new candidates are generated to replace the original, one with the potential reduplication noted and the other treating the entire set of root syllables as a single homogeneous unit. Thus, if the original input is [*sə kalikali*] 'never at all', one of the candidates passed on to the reduplication check will be [[*sə* prefix] [*ka li ka li* root]]. The reduplication check, in turn, will generate two possible structures underlying this input.

(18) [[*sə* prefix] [*ka li ka li* root]] ⟶

a. [*sə* prefix] + [REDUP–*ka li* root]
b. [*sə* prefix] + [*ka li ka li* root]

The second routine treats words in which two sequences of root syllables occur, separated by some affixal material. Such candidates might represent cases of reduplicated morphologically complex stems, like *sə kali–sə kali* 'from time to time', in which a prefix intervenes between the two sequences of root syllables. Words like *tulismə nulis* 'write: reciprocal' also fall into this category, with a prefix surrounded by two copies of the root. For these two-root cases, the routine checks to see exactly what sort of syllables intervene. Suffixes are classed as belonging to the first root; prefixes as belonging to the second. The candidate is thus broken into two halves, allowing the matching routine to compare like quantities of formatives:

(19) [*sə* prefix] [*ka li* root] [*sə* prefix] [*ka li* root] ⟶

a. [*sə* prefix] [*ka li* root] — [*sə* prefix] [*ka li* root]
b. [*sə* prefix] [*ka li* root] [*sə* prefix] [*ka li* root]

In (19a), for example, the prefix and root which precede the dash are compared to the prefix and root which follow it. As in example (18), a match generates two candidate parses, one with the reduplication (19a) and the other without (19b). A failure at this point triggers a subroutine which checks specifically for 'quasi-infixed'

cases like those described in section 5.2.6. If the material intervening between the two roots is one of the prefixes which participates in these structures, then the two roots are compared directly to see whether they are identical. If so, then two candidates are generated, one with the special form of reduplication noted, the other with the two roots unchanged.[10]

In contrast to standard reduplication-checking mechanisms, which arbitrarily compare n–segments with n–segments, the reduplication check described here compares structurally identical morphological *constituents* (i.e., root with root, prefix+root with prefix+root, and so on). This approach is clearly warranted for Indonesian and other languages with this sort of full reduplication: morphological constituents are copied by the reduplication process, not arbitrarily long strings of segments. Besides being more plausible on psychological grounds, this principled method of checking for reduplication is also far more efficient than the alternative.

5.4.5 Root Search

The final result of the phonological preparse, then, is an exhaustive list of all potential underlying formative combinations for an input word. These hypotheses range from the default hypothesis — that the input syllables represent a single undifferentiated root — to the most complex possible interpretation of the phonetic information. The number of hypotheses generated for each input word will depend on how many of the input word's syllables can be matched against the set of affix-shapes, and whether or not a phonological reduplication is present.

The preparse component makes no attempt to bracket the word's formatives into any particular structure, and up to this point no lexical lookup has yet been performed. Although the affix-stripping procedure is akin to lookup, the two differ crucially in that no gloss or semantic effect is associated with identifying potential morpheme boundaries; this requires access only to a list of phonological shapes. The root lookup routine which follows, on the other hand, entails retrieving the semantic, syntactic, and subcategorization information associated with each root — in other words, full access to the lexicon. Only after the syntactic and semantic class of the root is identified can the specific meaning of an individual affix or process morpheme can be identified.

[10]This rather motley assortment of reduplication-checking devices is suspect from a psycholinguistic point of view; in some sense all of these different sorts of reduplication represent reflexes of the same process and ought to be discovered by a single process. The reduplication check could be made far more elegant if it were given access to stress information about a word, since, as Cohn (1987) notes, stress contours as well as segmental information are copied by Indonesian reduplication. Finding two identical stress contours over a single word, for example, might trigger a comparison of the formatives/syllables dominated by these contours.

The lexicon consists of simple roots and those morphologically complex forms which behave idiosyncratically. The root search procedure, which is essentially dictionary lookup, produces the syntactic and (when relevant) semantic class to which a candidate's root belongs. This step also serves to doom a great number of incorrect parses, since their putative roots do not exist in the lexicon.

Irregular Lexical Items

Some of the irregular forms discussed in section 5.3.3 are treated by lexical lookup. Apparently complex forms like [*gi gi*], 'tooth' — which are in fact monomorphemic roots — are identified as wholes by the lexical search. Meanwhile, candidates which represent the normal morphological decomposition of such roots will fail, since the corresponding simple root does not exist in the lexicon. For example, given the input [*gi gi*], the phonological preparse will produce hypotheses both with and without reduplication:

(20) a. REDUP–[*gi* root]
b. [*gi gi* root]

For this particular word, (20a) will fail root lookup, since no root *gi* exists in the lexicon. (20b) will succeed, glossed as 'tooth'. Since reduplication has already been checked, there is no need to lexically mark such forms as irregularly reduplicated.

The question of exactly how Indonesian roots like *gigi* 'tooth' and *biribiri* 'sheep' should be represented in the lexicon is an interesting one: should they be treated as obligatory reduplications of the roots *gi* and *biri*, producing phonologically regular but semantically irregular effects? It is not entirely clear that these represent polymorphemic forms which have become lexicalized; they may only accidentally resemble reduplicated roots. In any case, their failure to feed the reduplication rule suggests that the constraint against re-reduplication is motivated by phonological, rather than morphological or semantic factors. Additionally, the fact that both the reduplicated and monomorphemic candidates exist during the course of the derivation might be used to explain why *gigi–gigi* is impossible: just as semantically ambiguous words in English seem to prime all potential readings simultaneously, these monomorphemic Indonesian "reduplications" might be claimed to (briefly) activate both the monomorphemic *gigi* parse and the reduplicated *gi* parse. At some point in the parsing process, speakers note the phonological reduplication, even though this parse is ultimately unsuccessful. Thus, any candidate like [REDUP–[REDUP–*gi*]] can be ruled out immediately because it violates Indonesian's constraint against re-reduplicating forms. In addition, an analysis such as*[REDUP–[gi gi]] must be

blocked by an extension of the same constraint to include monomorphemic forms which mimic reduplications.

An Example of Root Search

A typical example of root search is presented below. Suppose the syllabified phonetic input is:

(21) [bər kə li a ran]

Possible underlying shapes corresponding to this phonetic input are:

(22) a. [*bər* prefix] [*kə* prefix] [*liar* root] [*an* suffix]
b. [*bər* prefix] [*kə* prefix] [*liaran* root]
c. [*bər* prefix] [*kəliar* root] [*an* suffix]
d. [*bər* prefix] [*kəliaran* root]
e. [*bərkəliar* root] [*an* suffix]
f. [*bərkəliaran* root]

Once the root search has been performed, forms (22b–f) will fail as potential candidates, since *liaran*, *kəliar*, etc. do not exist in the root lexicon. In one step, then, the cohort of possible formative combinations for this input is slashed dramatically. The lexical entry for the successful root, *liar* 'free, wild' provides the necessary information about syntactic category and a gloss. This information, [Adj, 'free, wild'], is substituted for the phonological sequence [*liar*].

5.4.6 Morphological Matching

The final stage, morphological template matching, will thus treat only candidate (22a), the sole survivor of the root search. I will follow this candidate in some detail as it moves through the parsing process, in order to demonstrate just how the system arrives at a parse. The formatives in (22a) might represent several different morphological bracketings:

(23) a. [*bər* [*kə* Adj] *an*] ([*kə* Adj] circumfixed with *bər+an*)
b. [[*bər* [*kə* [Adj *an*]]] ([Adj *an*] prefixed with *bər* and *kə*)
c. [*bər* [*kə* Adj *an*]] (Adj circumfixed with *kə+an* and prefixed with *bər*)

In order to find and decide among these competing structures, a bottom-up matching routine attempts to build up stems by combining constituent formatives

into a single stem. Each candidate parse is checked against a total of 27 morphological patterns during the initial template-matching stage. This routine iterates, gradually building up complex stems by adding formatives to the root on each pass through the template-matcher. The templates, which take the following form, are listed in Appendix B:

(24)

prefix	+	Adj
VTroot	+	suffix
VT	+	suffix
circumfix	+	VTroot

The 27 templates represent combinations of different stem-types with the basic morphological processes which build Indonesian words: prefixation, suffixation, circumfixation, and reduplication. Each time a candidate parse or some subpart of a candidate matches one of these templates — for example, when a sequence of any prefix followed by an adjective exists, the appropriate word-formation rule is triggered (Appendix C lists these rules). For example, if the prefix encountered in the structure 'prefix + Adj' is *sə*, rule 19 will apply, combining the two constituents into an adjectival stem meaning 'as Adj as.' If the prefix is *pəN*, then rule 20 is activated to create an intransitive verb stem meaning 'to be characterized by/do Adj.'

Also represented in the set of morphological templates are those word-shapes which have some idiomatic, or non-compositional, effect on the syntactic/semantic features of the word (section 5.2.3). A shape such as

[REDUP–Adj+*an*]

might be expected to represent a nominalized, intensified adjective, according to the general set of morphological rules in Appendix C. However, this particular word structure seems to have special status in Indonesian, producing adjectives of somewhat weakened intensity, as with English *-ish*. Similarly, 'infixing' reduplications like those described in section 5.2.6 produce forms connoting 'reciprocality,' an effect which cannot be made attributable to the order in which the word's formatives are combined by the template-matching procedure. Such cases are treated by matching whole-word templates against candidate formative-lists. Any time such a sequence of formatives is encountered, the appropriate 'idiomatic' rule is triggered. This set of idiomatic rules is ordered before the normal rule set, with the two sets applying disjunctively on each pass that a candidate makes through the morphology. Matching one of the templates that produces a semantically irregular effect prevents further attempts to parse a candidate's formatives, thus preventing regular but incorrect parses for such forms.

Each time constituents of a candidate parse are combined into a single stem, that candidate is altered to reflect 1) the new syntactic/semantic category of the stem, and 2) whatever formatives remain unmatched by the template. The candidate's stem and remainder pass through the set of templates iteratively, until either all the candidate's formatives are successfully combined into a single stem, or the matching procedure fails (i.e., when no rule exists to unite the current stem with any of the candidate's remaining formatives).

As with the phonological parse, each time multiple matches are generated for a single candidate, the list of candidates under consideration must be increased. For example, the candidate

[*bər* prefix] [*kə* prefix] [Adj] [*an* suffix] ('free, wild')

matches three structural templates on its first pass through the matcher: the root might be prefixed with *kə*, suffixed with *an*, or it might be circumfixed with *kə+an*. Accessing the rule component will cause the first hypothesis to fail, since no there is no morphological rule which combines a prefix *kə* with adjectives. However, both the suffix and circumfix hypotheses are possible. The suffix *an* combines with Adj to create noun stems, as does the circumfix *kə+an*. After the first layer of affixation has been added to the root, then, two new candidates, each with a nominal stem and a remainder of unassociated formatives, have been created from the original:

(25)
a. [*bər* prefix]+[*kə* prefix]+[N]
(where N is derived from Adj+*an*: rule 10)
b. [*bər* prefix]+[N]
(where N is derived from Adj+*kə+an*: rule 6)

The matching process iterates, combining more and more potential layerings of formatives into homogeneous stems on each pass. In any case, candidate (25a) fails on its next transit through the template/rule set, since *kə* cannot combine with nouns (in other words, no rule exists to unify this prefix with an 'N' stem-type).

Candidate (25b), however, matches a rule which combines the prefix *bər* with nominal stems, creating an intransitive verb. Since combining this prefix with the stem exhausts all the putative formatives in the candidate under consideration, the parse is successful. The final result of the matching procedure then, is the stem-type, 'VI', and any cumulative semantic effects that the application of each rule has had on the core gloss in the lexical entry. Adding *bər* to a noun stem creates intransitive verbs which mean something like 'to do/have quality of X'. The nominal which was created by circumfixing the adjective 'free/wild' with *kə+an* is thus turned into a verb by the addition of the prefix *bər*.

5.4.7 Output

The final output of the parser includes a syntactic category, a gloss for the root, and where necessary (i.e., when a rule's operation has triggered it), a formulaic gloss into which the stem's meaning can be inserted (as is the case for the prefix *bər*). Thus, the final parse output for the input word [*bər kə li a ran*] consists of [VI, 'to do/have the quality of: wildness/freedom'].[11] Additionally, the few inflectional categories that Indonesian marks morphologically must also be included in the output. Thus, [*oraŋ–oraŋ*], a reduplicated nominal, is glossed to reflect the inflectional value of this reduplication: [N, 'human being', plural].

5.4.8 Philosophy, Generalizability, and Psycholinguistic Concerns

The parser discussed above operates serially, first creating multiple possibilities for the set of underlying formatives in the input word, and then sorting through these possible sets one by one to check for reduplication, looking up each candidate's root, and finally trying to assemble a grammatical word out of each candidate's component formatives.

The philosophy underlying this model is a desire to 'undo' not just any phonological rules which might have applied during a word's derivation, but also the cycles of affixation/morphological processes which built the stem outward from the root. Any time the set of morphological rules is able to successfully generate a word, using all and only the 'raw' formatives provided in the candidate, a successful parse is produced. Thus the same set of morphological rules used to generate words is used for parsing, a pleasing economy.

This model is not quite in keeping with a model of morphology and phonology like that of lexical phonology, as the effects of morphophonological and phonological rules are not undone cyclically. Instead, a brute force approach is used to find all possible traces of phonological rules which might have operated in deriving a surface form, with potential bracketings for each candidate being established only after this phase. This appears to be an unavoidable problem in reversing a generative process like that proposed by lexical phonology, however, since that theory's cyclic bracketing erasure eliminates exactly the sort of boundaries which are crucial to a successful morphological parser's operation.

[11] This word is variously defined by dictionaries as 'roam free', 'lounge/hang about'. Although the parse output is not exactly identical, it is similar to these glosses.

Efficiency

The manner in which the parser generates more and more candidates, each of which must be checked as a possible correct parse; might seem computationally inefficient. An exponential explosion in the number of candidates parses is equally undesirable from a psycholinguistic point of view. Furthermore, the serial application of rules to a potentially very large set of candidate underlying forms appears to be a serious drawback to the model from a psycholinguistic standpoint, for it suggests that all else being equal, a word whose phonetic string might represent many possible morphological structures could take longer to process than a word with fewer such candidate structures. (There is, as one might expect, no evidence to support such a claim.)

However, neither the threat of a computational explosion nor the differential processing times for words with few/many candidates is a genuine problem for the parsing model I have presented. First of all, the expected explosion of candidate forms never occurs: although a large number of candidate-parses may be generated for a given word during the initial phonological preparse, this number is immediately reduced drastically by the root-lookup procedure, which strikes from the set any candidates whose roots do not exist in the lexicon. In practical terms, even the most complex words usually have no more than ten candidates by the time morphological template matching begins.

The size of the phonological-parse cohort of candidates could have been further constrained by checking the list of roots each time a new syllable is added to each candidate's root. If at any time the addition of a new syllable creates a sequence of syllables which does not exist in the root lexicon (perhaps represented by a finite state automaton, with each arc representing a syllable), that parse would fail. Since a single incorrect parse can spawn numerous progeny — all also incorrect — forcing a failure early on in the candidate-generation process would save a good deal of unnecessary parsing later on.

However, the usefulness of such a procedure must be balanced against the computational overhead involved in its implementation. For Indonesian, the complexity of this additional machinery would not justify the negligible improvement in efficiency it might yield.

Although implemented serially, this parser in fact attempts to model a parallel processing algorithm for word perception in which potential parses are hypothesized and tested simultaneously. Given proper computer architecture, the sort of scheme described above could be implemented in a truly parallel fashion. A parallel implementation would prevent the exponential growth in processing time observed in a serial implementation: humans simply do not take longer to process words with

multiple possible underlying forms than those which are unambiguous. Simultaneously processing a large number of hypotheses might be somewhat slower than a parse which needs to test only one hypothesis, but this growth in processing time will not approach the sort of exponential explosion observed in serial/backtracking routines as the search space grows larger.

Appendix A: Productive Affixes

Prefixes	Suffixes	Circumfixes
məN	an	kə+an
pəN	kan	sə+an
di	i	pər+an
bər		bər+an
pər		pəN+an
tər		
sə		
kə		

The possessive enclitics *mu*, *ku*, and *ña* also combine productively with stems.

Appendix B: Templates

Candidates are matched against these templates as they move through the morphological component. Each time a template is matched, the rule associated with its relevant formative is called up, adding this new formative to the stem. The first four templates form a block which is ordered disjunctively with the other rules. These four templates produce non-compositional effects, and the normal stem-building rules must be prevented from applying in these cases.

If a candidate doesn't match one of these four 'idiomatic' templates, it may match one or more of the shapes in the second block of rules. If more than one template is matched, the candidate is doubled, with a separate new candidate for each possible combination of formatives.

Subscripts indicate either morphological or semantic features which are referred to by particular morphological rules. The subscript *root* denotes a bound verb root, which must be distinguished from derived stems. *VI* refers to an intransitive verb stem and *VT* to a transitive verb stem.

Idiomatic Structures Block

Templates:			Applicable Rules:
prefix	+	REDUP II–VI_{root}	1 ('infixed' prefix)
prefix	+	REDUP II–VT_{root}	1 ('infixed' prefix)
sə+an	+	REDUP–N_{time}	11
REDUP–Adj	+	an	35
REDUP–N	+	an	36

Regular Rule Block

Templates:			Applicable Rules:
Prefix	+	VI_{root}	
pəN			21
tər			22
məN			20
REDUP			17
Prefix	+	VT_{root}	
tər			23
məN			5
di			24
REDUP			17
Prefix	+	N_{number}	
kə			25
məN			20
pər			28
REDUP			29

Templates:			**Applicable Rules:**
Prefix	+	N_{mass}	
sə			27
REDUP			10, 29
Prefix	+	N_{count}	
sə			26
REDUP			29
Prefix	+	N	
bər			21
REDUP			10, 29
Prefix	+	Adj	
pəN			21
tər			18
məN			19
sə			19
REDUP			37
VT	+	Suffix	
		an	15
		kan	16
		i	16, 17
VT_{root}	+	Suffix	
		kan	8
N_{number}	+	Suffix	
		an	12

Templates:			Applicable Rules:
N_{Arabic}	+	Suffix	
		i	18
N_{place}	+	Suffix	
		kan	5
N	+	Suffix	
		i	5
Adj	+	Suffix	
		an	7
		kan	8
VI	+	Suffix	
		kan	8
$VI_{root-stative}$	+	Suffix	
		i	8
		kan	8
VI_{root}	+	Suffix	
		i	14
		kan	5, 14
$VI_{root-feeling}$	+	Suffix	
		i	5
Adj	+	Circumfix	
		kə+an	6, 7

Templates:			Applicable Rules:
N	+	Circumfix	
		kə+an	6, 7
		sə+an	9
		kə+ña	12
VT	+	Circumfix	
		kə+an	13
		pər+an	3
		sə+ña	9
		bər+an	1
VI	+	Circumfix	
		kə+an	4
		bər+an	2
VI_{root}	+	Circumfix	
		pər+an	3, 2
		pəN+an	3

Appendix C: Rules

The following rules specify the syntactic, and in many cases, semantic effect of adding a particular formative to a stem.

1	⟶	V	(reciprocal/variety)
2	⟶	VI	(disorderly group activity)
3	⟶	N	(action)
4	⟶	VT	(antipassive)
5	⟶	VT	
6	⟶	N	(abstract)
7	⟶	Adj	(intensified)

8	⟶	VT	(causative)
9	⟶	Adv	(manner)
10	⟶	N	(collection)
11	⟶	Adv	(duration: 'all N long')
12	⟶	N	(number: 'all # of them')
13	⟶	V	(ability)
14	⟶	V	(imperative)
15	⟶	N	(patient, product)
16	⟶	V	(allative)
17	⟶	V	(iterative/intensive)
18	⟶	Adj	
19	⟶	Adj	(same degree)
20	⟶	VI	
21	⟶	V	('do/be characterized by X')
22	⟶	Adj	(connotes accidentalness)
23	⟶	Adj	(passive deverbal)
24	⟶	V	(passive)
25	⟶	Adj	(ordinal)
26	⟶	N	('only one N')
27	⟶	N	(predicative)
28	⟶	N	(fraction)
29	⟶	N	(plural)
30	⟶	Adv	(time: 'at X')
31	⟶	V	(neg: 'still')
32	⟶	N	
33	⟶	N	('all # of them')
34	⟶	Adv	('N by/after N')
35	⟶	Adj	('-ish'/lessened intensity)
36	⟶	N	(toy)
37	⟶	Adj	(distributive)

Chapter 6

Do People Parse?

Karen D. Emmorey

Psycholinguists generally agree that syntactic representations are constructed on-line during language comprehension, although there is much debate about the nature of these representations and how they are constructed. Since all possible sentences could not be stored in memory, syntactic representations are assumed to be computed. Not only are syntactic rules recursive, thus allowing for the possibility of infinitely long sentences, but the number of possible combinations of words forming a sentence is astronomical. Therefore, we parse sentences. But do we parse words? That is, are morphological representations of words constructed on-line? The arguments put forward for syntactic parsing do not apply as convincingly to morphology. It is possible to conceive of storing all existing words in the lexicon — in fact, this has been proposed for English (Butterworth (1983)). Furthermore, the trade-off between storage and processing costs must be considered more seriously. It may be more economical to access and store entire words rather than to store only stems or roots and require mandatory parsing for each encounter of a complex word.

However, the question to be explored in this paper is not whether the internal morphological structure of words is represented in the linguistic system or in a psychological processing system. The linguistic and psychological evidence shows clearly that speakers are aware of the morphological structure of words. Speakers can provide judgments and intuitions about the morphological relatedness of words in their language, and this data can be used to construct linguistic theories of morphological representation. Speakers are also able to create and understand new words formed according to the rules of their language (see for example the famous "wug"

experiment; Berko (1958)). Furthermore, several psycholinguistic experiments indicate that the morphological relations between words are represented in the lexicon (Kemply and Morton (1982), Stanners et al. (1979)).

The question addressed here is whether speakers construct on-line morphological representations during language comprehension. To answer this question explicitly, it is important to distinguish results which indicate on-line morphological computation from results which show only that morphological relationships are represented in the lexicon or that speakers can use a morphological rule in generating or interpreting *new* morphologically complex words. For example, priming experiments have shown that the lexical decision time to a word preceded by a morphologically related word (*sings–singer*) is faster than when it is preceded by an unrelated word (*singe–singer*) (Murrell and Morton (1974)). Without certain theory internal assumptions, these results can only be taken to reflect the internal organization of the lexicon and not morphological analysis prior to word recognition.[1] Experiments which allow subjects unlimited time to complete a task cannot be used in determining the nature of on-line morphological processing because their performance may reflect several effects due to strategies or other knowledge which may not be used during normal language comprehension. It is of crucial importance to determine the nature of the representations that subjects are constructing in *real time* (for a good discussion of the theoretical and methodological issues involved here see Tanenhaus et al. (1985)).

6.1 Evidence for Parsing

An initial caveat is required before evaluating the psycholinguistic evidence bearing on morphological parsing. Except where noted, all of the research cited below has been conducted with English. Any conclusions reached here should be confined to the parsing of English morphology until more experiments can be done using other languages, preferably those with richer morphological systems. Furthermore, most of the psycholinguistic research on morphological processing has been conducted with visually presented materials. The psychological processes involved in reading may be quite different than those involved in auditory language perception (see Marslen-Wilson (1984)). Caution should be used in extending any of these results

[1] Priming facilitation is interpreted within a logogen model (Morton (1969)) to reflect lowered thresholds for recognition units due to lexical access. Varying facilitation is predicted depending on whether these units correspond to morphemes or words. However, morphological priming facilitation may simply reflect the connections between lexical representations, similar to network explanations for semantic priming. The logogen model is forced to assume that the facilitation reflects the nature of recognition units because no connections between units are hypothesized.

to the auditory domain.

6.1.1 Morphological Complexity

If morphological parsing occurs during word recognition, one might hypothesize that polymorphemic words should take longer to recognize than monomorphemic words. Polymorphemic words may have to be decomposed into their components whereas no such analysis is required for monomorphemic words. Both poly and monomorphemic words may be submitted to the same analysis procedure, but if affixes have to be stripped from the complex word to locate the root, this procedure might be expected to take some processing time.

The lexical decision task is the tool most often used in psycholinguistic research. In a lexical decision task, subjects are presented with real words and nonwords and are asked to decide as quickly as possible whether a given item is a real word. The reaction times of subjects are measured and are hypothesized to reflect the time for a subject to access and recognize a word in the lexicon. There are some problems with considering lexical decision time as a "pure" measure of lexical access time (see Seidenberg et al. (1984)), but this issue will not be addressed here.

In any case, lexical decision experiments have revealed that with visual presentation bimorphemic words do not take longer to recognize than monomorphemic words (Henderson et al. (1984); Taft and Forster (1974) — cited in Cutler (1983)). In contrast, Emmorey (1986) found that with auditory word recognition, lexical decision times for suffixed words were longer than for monomorphemic words. On the other hand, reaction times for prefixed words were found to be *faster* than for monomorphemic words. Apparently, there is not a positive correlation between the number of morphemes and processing time for auditory word recognition. Emmorey (1986) proposed that lexical decision times were faster for prefixed words because subjects recognized the prefix early and were thus biased to respond "word" more quickly in the lexical decision task. Emmorey (1986) further argued that the lexical decision delay for suffixed words was due to the additional time required to recognize the suffix.

At least for auditory word recognition, there is some evidence that the morphological structure of words is recognized on-line and can affect word recognition. For visual word recognition simply comparing reaction times for poly and monomorphemic words does not reveal effects of morphological complexity. However, other avenues of research are available that bear on the affix stripping hypothesis for visual word recognition.

6.1.2 Pseudoaffixation Effects

Pseudoaffixation effects refer essentially to the possible effects of false parsing. Words which appear to be affixed but are not (e.g. *relish* vs. *revive* when presented visually) may take longer to recognize because they are initially misparsed. Taft and Forster (1975) propose a model of the lexicon in which prefixed words do not have separate entries but are accessed by their roots. According to this model, prefixed words are recognized in the following manner:

1. Recognize that a prefix is present and remove it for lexical search.
2. Search the lexicon for the root.
3. When the root is found recombine it with the prefix to form the word. At this point recognition occurs.

This procedure is based on evidence which indicates that nonwords derived from prefixed words (e.g. *vive* from *revive*) take longer to reject in a lexical decision task than nonwords derived from pseudoprefixed words (e.g. *lish* from *relish*). Taft and Forster (1975) suggest that the root *vive* is initially found in the lexicon but later rejected as a nonword. The nonword *lish* can be rejected immediately, however, since there is no root *lish* represented in the lexicon. Taft (1979) extends this prefix stripping model to suffixes.

The primary hypothesis that arises from the Taft and Forster model is that pseudoaffixed words should take longer to process than true affixed words because they are initially misparsed. When no "root" is found in the lexicon for the pseudoaffixed word, a second search must be initiated for the whole word causing a delay in recognition. Note that due to this initial misparse pseudoaffixed words should also take longer to process than monomorphemic (nonaffixed) words.

However, results comparing pseudoaffixed words with true affixed words and monomorphemic words have been generally negative. Neither Manelis and Tharp (1977) nor Henderson et al. (1984) found that pseudosuffixed words (e.g. *somber*) took longer to recognize than true suffixed words (e.g. *sender*). Segui and Zubizarreta (1985) also found no reaction time difference between pseudoaffixed and true affixed words in French. Henderson et al. (1984) found that neither pseudoprefixed words nor pseudosuffixed words took longer to recognize than nonaffixed monomorphemic words. Similar results have been found with auditory presentation (Emmorey (1987)). In contrast, Taft (1981) found that naming latencies for visually presented words (i.e. time to begin reading a word aloud) were longer

for pseudoprefixed words than true prefixed words or words with no prefix. However, Henderson (1985) points out several flaws in this experiment, and its relevance is thus uncertain.

It seems clear from the above results that if words are decomposed morphologically prior to lexical access, this process is not misled by false parses. Or rather, words with potential misparses do not incur processing costs. We will discuss below the hypothesis that pseudoaffixed words are indeed misparsed, but a search for the nonanalyzed form (e.g. *somber* or *tester*) also occurs *in parallel* with the lexical search for the root of the decomposed form (e.g. *somb* or *test*).

6.1.3 Other Effects of Morphology

Letter search/cancellation tasks have been shown to be sensitive to the morphological structure of words. Smith and Groat (1979), Drewnowski and Healy (1980), and Smith (1982) have shown that in searching through prose for a target letter, subjects are more likely to miss a target that is part of an inflectional suffix. Smith and Groat (1979) found that the rate of omission for *e* in *-ed* was higher than when *e* appeared in a nonaffixed word (e.g. *hundred*). Drewnowski and Healy (1980) also found higher omission rates for the letter *n* in the inflectional suffix *-ing* compared to words ending in a nonaffix string (e.g. *wing* or *sterling*). These results strongly suggest that words are broken down into their morphological constituents during reading.

One hypothesis put forth by Drewnowski and Healy (1980) to explain the omission errors is that suffixes are read as separate units, and not broken down into letters. Another possibility, however, is that more errors occur because inflectional suffixes are less "salient" and not because they are read as an unanalyzed unit (although this might occur). That is, omissions may occur because subjects ignore inflectional suffixes during their search since they do not provide semantic information for the interpretation of the word or sentence.

These two hypotheses can be tested in English with the regular plural which can appear simply as the letter *s*. If the reason for overlooking letters in inflectional suffixes during cancellation tasks is that suffixes are not broken down into letters, then one would expect no difference between a plural *s* (e.g. *hens*) and an *s* that ends a word but is not a plural suffix (e.g. *lens*). The plural *s*, of course, cannot be broken down any further. The unit itself is the target in the search task, and therefore should be just as detectable as a nonplural *s*. On the other hand, if letters are omitted because inflectional suffixes are less salient in the semantic structure of a text, one would predict that the plural *s* would be omitted more often that the nonplural final *s*.

Preliminary results from such an experiment reveal that subjects do not make more cancellation errors on plural *s* compared to word final *s*. Ten subjects read a 550 word passage and were asked to cancel all occurrences of *s*. Subjects failed to cancel 5% of word final *s*'s and 4% of plural *s*'s. Apparently, subjects do not make more errors on the plural inflection since the *s* can be read as a unit which itself is the target letter. The hypothesis that inflectional suffixes are not detected in the search task because they are semantically less salient is not supported. The letter cancellation results reviewed above indicate that during reading the morphological structure of inflected words is recognized and inflectional suffixes are read as a unit. Psychological theories about how this structure might be recognized and/or parsed are discussed next.

6.2 Evidence for how people parse

6.2.1 Parallel Processing

As noted above, pseudoaffixed words do not take longer to process than true affixed or monomorphemic words. Based on these findings, Henderson et al. (1984) proposed that the Taft and Forster prefix stripping model be altered to contain a parallel search for the root of the decomposed words and the whole word. A parallel search predicts no difference between pseudoaffixed words and monomorphemic words because the search for the whole word will succeed for both word types in the same amount of time (all things being equal). The false parse of the pseudoaffixed word will simply fail and will not slow down the parser as it did in the earlier model which required a second search for the whole word. This modification maintains the original hypothesis of Taft and Forster that the roots of prefixed (and suffixed) words are stored in the lexicon, and complex words can only be recognized by prelexical morphological decomposition.

Caramazza et al. (1985) propose a more detailed and slightly different parallel processing model based on data from Italian. Caramazza et al. propose that morphologically decomposed representations are stored in the lexicon as logogen-like recognition units (see Morton (1979)). This model assumes two lexical access mechanisms: 1) a whole-word address system that, in the case of *known* morphologically complex and monomorphemic words, accepts the entire letter string as input and matches it with its corresponding recognition unit, and 2) a morpheme address system whose recognition units correspond to morphemes instead of whole words.

Caramazza et al. assume that because the morphological parsing procedure is a complex process relative to the whole-word address procedure, it is also a relatively slow process. Consequently, the effects of the morphological parsing procedure are

not revealed in the lexical access of complex words since their lexical representations are addressed more efficiently through the whole-word procedure. Thus, the model explains the lack of recognition time difference between pseudoaffixed words, true affixed words, and monomorphemic words when presented visually: all are accessed through the whole-word address procedure. The nature of the whole-word procedure is very unclear, and we will return to this point in a moment. First, the evidence for the hypothesized morphological parsing procedure should be presented.

Since the effects of the morphological parsing procedure for words are masked by the more efficient whole-word procedure, support for the parsing procedure is based entirely on evidence from normal subjects' and dyslexic patients' processing of *nonwords*. Laudanna and Burani (1985) report that Italian normal subjects took much longer to reject nonwords that contained real root or suffix "morphemes" in a visual lexical decision task. For example, nonwords made up of two morphemes (e.g. *cantevi: cant-* "to sing" and an inappropriate inflection *-evi*) took longer to reject than nonwords with only one morpheme (e.g. *canzevi; canz* is not a real root). Both of these nonword types took longer to reject than nonwords which contained neither a root nor a suffix (e.g. *canzovi*). Laudanna and Burani explain this pattern of results as attempts of the morphological parsing system to analyze the nonwords as words. Caramazza et al. (1985) present a similar explanation for the errors of a dyslexic Italian patient reading the same nonwords types. Experiments with English have also shown longer lexical decision times for nonwords with "morphemes" (e.g. *desker, garnly*) than for those without (e.g. *garpod*) (Emmorey (1987), Henderson et al. (1984), Manelis and Tharp (1977)). Evidence from nonword lexical decision experiments strongly support a morphological parsing procedure.

However, Caramazza et al. (1985) propose that *known* complex words are processed by activating a whole-word entry which addresses a morphologically decomposed orthographic lexicon. The orthographic lexicon is distinct from the lexical address system which consists of whole word "addresses" that become activated in a logogen-like manner (Morton (1979)). An activated whole-word address specifies a morphologically decomposed root morpheme and affix representation in the orthographic lexicon. For example, the word *walks* activates an address unit corresponding to the whole word *walks* which serves to access the root *walk* and the verbal suffix *-s* in the lexicon. However, if this whole-word procedure is how words are normally accessed, much more must be said about how the whole-word units come to access the stored decomposed representations. In addition, proposing such a procedure means that words are represented twice in the processing system — once as a whole word unit in the address system and once in a decomposed form in the lexicon. Surely, this is an undesirable result.

The evidence provided to support the whole-word access procedure is based

on word frequency experiments. Results from Italian (Burani et al. (1984)), English (Taft (1979)) and Serbo-Croation (Lukatela et al. (1980)) indicate that the frequency of both the root and the frequency of the surface (i.e. inflected or derived) form affect reaction time in lexical decision tasks. Words with high frequency roots are recognized faster than words with low frequency roots, although their affixed forms are matched for frequency. However, if words are matched for *root* frequency, affixed words with high frequencies are recognized faster than affixed words with low frequencies. For example, *thing* and *world* have the same root frequency, but *things* is more frequent than *worlds*. Lexical decision time for *things* is faster than for *worlds* (Taft (1979)).

Burani et al. (1984) account for the frequency effect of affixed words by assuming that the whole-word address units are activated in a logogen-like fashion and that their response thresholds are influenced by the number of times these units have been activated. The response threshold for a whole-word unit is lowered each time the affixed form of the word is encountered. Lowered response thresholds lead to faster recognition times (Morton (1969)). For example, every time the whole-word address unit for *walks* is activated, it leads to a lowering of the unit's response threshold. A separate whole-word unit exists for *walked*, and its response threshold is lowered whenever *walked* is encountered. The effects of root frequency on reaction time are accounted for by the additional presence of the root representation (i.e. *walk*) in the orthographic lexicon.

An alternative account may be possible, however, which does not depend on separate whole-word units. The frequency effect for morphologically complex words may be due to the *connection* between roots and affixes in the lexicon. Caramazza et al. (1985) propose that information about permissible affixes is stored with the root, and it is proposed here that this information takes the form of network-like connections between roots and their permissible affixes. With each encounter of a complex word, the connection between the root and the affix is strengthened, and the strength of these connections affects the lexical decision time for complex words. The effect of root frequency can be accounted for by assuming that each encounter of the root lowers the logogen response threshold.

Problems for this connectionist account (and especially for the whole-word address account) arise when languages with complex morphology are considered. When the number of possible root/affix orderings and combinations is large, a connectionistic account becomes unwieldy. However, the spirit of the proposal may be salvaged in that the allowable root/affix combinations must be specified in the lexicon, and the frequency of these combinations is what affects the speed of word recognition. Therefore, the frequency effect should arise from the mechanism that provides this specification, not out of a separate address system in which all complex words are

stored redundantly as whole entries.

The remaining evidence in favor of the whole-word access route is essentially negative evidence; that is, the lack of response time differences between morphologically complex words and monomorphemic words for reading. Caramazza et al. assume that the parsing procedure requires more processing time; therefore, if parsing were the only access route into the lexicon, morphologically complex words would take longer to recognize compared to monomorphemic words, contrary to current findings (Henderson et al. (1984)).[2] If this assumption is abandoned, then the reason for the whole-word procedure as the normal access route for morphologically complex words disappears.

However, as Henderson et al. (1984) proposed, we still need a parallel search procedure for whole words to explain the lack of misparsing effects for pseudoaffixed words. This procedure differs from that proposed by Caramazza et al. in that morphologically complex words cannot be recognized by the whole word search — they must be decomposed. Only pseudoaffixed words and monomorphemic words are accessed by this procedure; again, the false parse of the pseudoaffixed word simply fails.

6.2.2 Temporal or Linear Constraints

Although we have repeatedly stated that little evidence for pseudoaffixation effects have been found, Manelis and Tharp (1977) did observe an effect of pseudosuffixation in a restricted case. In their experiment subjects were asked to make a double lexical decision to word pairs. Reaction times for homogeneous true suffixed pairs (e.g. *tester–sender*) were not longer than for pseudosuffixed pairs (*sister–somber*). However, reaction times for mixed pairs (*tester–somber*) were longer than for homogeneous pairs. Manelis and Tharp suggest that this result is due to semantic priming in the homogeneous pair. That is, the homogeneous pair is faster because the words share a semantic element (namely the suffix). But this explanation leaves unanswered why the pseudosuffixed homogeneous pair (*sister–somber*) should be responded to faster since these words share no semantic element.

To follow up on this result, Segui and Zubizarreta (1985) conducted a similar experiment in French. Homogeneous and mixed pairs of pseudoaffixed and true affixed words were presented to subjects. One member of the pair was presented first (the context word), and the subject simply read this word silently. The second member of the pair (the target word) was then presented for lexical decision. Interestingly, only pairs with a suffixed target produced a slower reaction time for mixed pairs (e.g.

[2]Preliminary results from a visual lexical decision task appear to indicate that some suffixed words do take slightly longer to recognize than monomorphemic words (Emmorey (1987)).

entier–potier) compared to homogeneous pairs (*rosier–potier*). Pairs with prefixed target words did not show a reaction time difference between mixed (*préfet–prénom*) and homogeneous (*préface–prénom*) pairs.

These results reveal an asymmetry in the processing of prefixed and suffixed words. While the processing time for suffixed words varied according to the morphological characteristics of the word context, this was not observed for prefixed words. When preceded by a suffixed word, suffixed targets were recognized more quickly than any other target word type. Furthermore, the recognition of suffixed but not prefixed words was facilitated when words with the same morphological structure but different roots and affixes were presented successively prior to the target affixed word (Colé et al. (nd); cited by Segui and Zubizarreta (1985)). That is, the recognition of suffixed words was facilitated when the preceding context words simply shared the property of being suffixed but did not share a particular suffix or root. Apparently, the parsing procedure itself can be facilitated. But why aren't lexical decisions to prefixed words aided by the same circumstances? One explanation is that prefixed words are stored as separate lexical entries, and thus the parsing procedure does not succeed for either prefixed or pseudoprefixed words and thus cannot be facilitated. For languages like French and English which have only a few productive prefixes, prefixed words may indeed have separate entries, but for languages with many productive prefixes, this hypothesis is probably not correct.

Another possible explanation depends on the assumption that linear constraints impose different lexical access procedures for prefixed and suffixed words. For suffixed words the root must be accessed prior to the suffix. Suffixes may be stored in a separate lexicon or with the root entry. Many investigators have proposed that (at least some) affixes are stored separately from roots (Bradley et al. (1980), Garrett (1980), Caramazza et al. (1985)). The results of Segui and Zubizarreta (1985) may be explained by hypothesizing that the recognition of a suffixed word is facilitated when it is preceded by a suffixed word because access to a suffix lexicon has been facilitated. Priming occurs even for words with different suffixes because the same parsing procedure is utilized to recognize both words, and it is this procedure that is facilitated. Facilitation does not occur when a pseudosuffixed word precedes a true suffixed word because the suffix lexicon is not accessed for the pseudosuffixed word. For pseudosuffixed words the root (i.e. the pseudosuffixed word itself) is accessed first and prevents access to the suffix lexicon.

In contrast, for both pseudoprefixed and prefixed words access to a prefix lexicon may occur initially. However, a parallel search of the root lexicon may also begin since the parser cannot determine if the word to be recognized begins with a prefix or with a similar word initial string (i.e. a "pseudoprefix"). For prefixed words, the prefix is recognized first, and then the root is accessed and recognized. For

pseudoprefixed words that were parsed as truly prefixed, no root is found, and this analysis fails. However, the parallel search that began in the root lexicon succeeds. Because the prefix lexicon is accessed regardless of whether the prefix string turns out to be a true prefix, lexical decision is facilitated for target prefixed words when preceded by either a true or pseudoprefixed word. Therefore, no reaction time difference is observed between these two word categories.

This analysis suggests that both parallel and serial processes are operating during lexical access for morphologically complex words. The above discussion also seems to suggest that prefix representations may be stored in a separate lexical subsystem from roots and suffixes. Initially, parallel access occurs in the prefix lexicon and the root lexicon. For pseudoprefixed words only the root search succeeds. For true prefixed words the prefix is first accessed in the prefix lexicon, and then the root is accessed. For suffixed words, of course the prefix lexicon search fails (unless the word is also prefixed), and the root search succeeds, and the suffix lexicon is then accessed. For pseudosuffixed words, the root lexicon search succeeds, and the suffix lexicon is never accessed (unless the pseudosuffixed word itself is suffixed, e.g. *sisters*). The parsing procedures and lexical representation system described here is only adequate for strictly concatenative morphology. Nonconcatenative morphology such as infixing, umlaut, ablaut, or semitic morphology pose great problems for all existing models of lexical access.

There is some evidence against a separate lexicon for suffix representations for auditory word recognition. Emmorey (1987) failed to find morphological priming effects for words that shared a suffix (e.g. *kindness–blackness*; *dancing–baking*), although priming was observed for words that shared a bound root (*permit–submit*). If the same suffix representation is accessed for words that share a suffix, priming should have been observed between the suffixed word pairs. Failure to find such an effect suggests that suffixes do not have separate representations that can be primed. It may be the case that for visual word recognition suffixes do have separate orthographic representations. Clearly, further research comparing visual and auditory word recognition for morphologically complex words is necessary.

Let us turn briefly to models of auditory word recognition where temporal constraints may be much more important. For a more detailed account of the role of morphology in auditory word recognition, see Emmorey (1987). Clear evidence from auditory lexical decision tasks indicates that listeners do not wait until the end of a word before lexical access begins. Reaction time is positively correlated with word duration when it is measured from word onset, indicating that the longer the word the longer the reaction time. However, reaction time is *negatively* correlated with word duration when it is measured from word offset (Jarvella and Meijers (1983), Emmorey (1986)). Since for longer words more time is available to process the word,

longer words produce faster reaction times when measured from word offset; hence, listeners must begin processing words before their offset.

Furthermore, several studies have shown that listeners are able to recognize words before hearing them completely. Fast speech shadowers can repeat back normal prose passages with a mean delay of 250 msec (Marslen-Wilson (1973)). Since the average word length is considerably longer, word recognition must have occurred very early. The gating task developed by Grosjean (1980) also indicates early word recognition. For this task subjects are given successively longer fragments of a word, and at each increment they are asked to say what they think the word is. Subjects could recognize words in context after hearing about 200 msec, although the total word duration was more that 400 msec. Other experiments utilizing word or phoneme monitoring techniques have also indicated that word recognition can occur very early. Although there is debate about the nature of lexical access mechanisms and representations, it is clear that lexical processing begins immediately and may be affected by temporal order.

6.2.3 The relation between psycholinguistics and computational linguistics

If a computational parsing system claims to model the mental processes involved in morphological analysis, the model must provide an account of how it relates to the psychological phenomena it is designed to explain. Of course, if the computational model makes no claims about psychological reality and simply attempts to parse morphologically complex words without regard to these concerns, no such account is necessary.

A computational system attempting to model psychological processes must address several issues concerning the level of correspondence between human and machine processing. The lowest level of comparison may be between neurons and circuits; however, machines and brains are obviously made of different stuff, and the details of mechanical operations may necessarily differ from neuronal functions. At a much higher level the correspondence may simply be between input-output functions (without regard to procedural method). Pylyshyn (1984) considers this comparison too weak for explanatory purposes and suggests the proper correspondence is somewhere in between these two extremes. He appeals to the intuitive notion of *algorithm* as the appropriate level to search for equivalence between programs and mental processes.

Is there any evidence that current computational parsers utilize algorithms similar to mental processes? Unfortunately, what we know about human morphological parsing based on psychological experiments may not be explicit enough to bear on

current computational models in any detailed way. For example, I know of no psycholinguistic evidence that would enable us to choose between a two-level model of morphological parsing (Koskenniemi (1983a)) and an analysis-by-synthesis model (Hankamer (1986)), although neither of these programs attempts to model human performance.

However, it may be informative and interesting to compare the current psychological evidence with these parsing models even though no claims of psychological reality are made. In either program, an increase in the number of morphemes should not incur processing costs, although word length will affect processing time as it does for human word recognition (Marslen-Wilson (1984)). As for pseudoaffixation effects, the KIMMO model (Koskenniemi (1983a)) predicts no effect of false parses because the lexicon acts as a filter eliminating false parses. Karttunen (1983) provides a fairly clear example based on orthography. If the KIMMO parser were attempting to parse the English word *changing* or *ringing*, the parser might attempt to find the possible roots *chang/change* or *ring/ringe*. If the parser operated serially, it might not discover that *chang* is not a possible English root until the end of the parse. However, because the recognizer only makes a single left-to-right pass through the word as it accesses the lexicon, the parser never attempts to analyze *changing* as *chang+ing* or *ringing* as *ringe+ing* since there are no entries for *chang* or *ringe* in the lexicon.

A similar explanation can be proposed for the Hankamer parser. For suffixed words, the root must be found first. The false "root" of a pseudosuffixed word is not found in the lexicon. Thus, *sister* is never analyzed as *sist+er* because *sist* is not stored as a root in the lexicon. It would be interesting to compare reaction times to pseudosuffixed words that contained "roots" that can be found in the lexicon (e.g. *lob* from *lobby*) and pseudosuffixed words whose "roots" cannot be found in the lexicon (e.g. *nast* from *nasty*). If *lobby* takes longer to recognize than *nasty*, the parser may be misled by the false "root" of *lobby*, but if no reaction time difference is observed we can only conclude that a serial search is probably not correct. As suggested above, the parser may still search for false "roots", but it does so in parallel with a search for the unanalyzed form.

A computational model of word recognition and morphological parsing that attempts to model psychological processes will have to contain certain features that are absent (to my knowledge) in current computational models. Word frequency greatly affects the accuracy and speed of word recognition and would have to be incorporated into any model of human performance. Furthermore, the letter search/cancellation data indicate that during reading affixes are read as units and not analyzed letter by letter. This type of recognition for affixes may have to be built into the parser.

6.2.4 Suggestions for further research

It is an open question how lexical processing systems vary as a function of the structural properties of their respective languages. Languages with different morphological properties need to be studied systematically to determine which aspects of human parsing are universal and which may vary with language type. Analogous to Universal Grammar, children may come equipped with a parameterized Universal Parser, and UG may be implicit in the parser or may interact with the parser as a separate system. The morphological properties of a language may determine how the Universal Parser is implemented. For example, if a language has no prefixes, the parser can search for the root without regard to possible prefixes, and although the parser may initially have this capability, it never becomes utilized. More sophisticated and interesting parameterized properties may be discovered by examining languages with different morphological systems.

Languages with productive morphological processes that cause phonological variation of the root or affix provide an important avenue for investigating lexical access mechanisms. Current models of auditory word recognition assume some type of matching procedure between the incoming stimulus and stored representations. Noncomputational passive matching is proposed for logogen-type models (Caramazza et al. (1985), Morton (1979)), but how word roots with different surface phonological forms are processed is not addressed. The passive matching approach itself is compatible with the two level parsing model of Koskenniemi (1983a), although the finite state automata for matching surface and underlying forms have not been proposed for psychological models. In the logogen model, passive matching occurs between a stored recognition unit (a logogen) and the word stimulus; the logogen "fires" and outputs a word response to a conceptual system as soon as enough stimulus information has been collected (Morton (1969)) — yet precisely *how* this information is collected is not made explicit. No evidence exists as yet to distinguish this passive model from a more active model of lexical access in which one underlying representation is stored, from which a surface form is computed and matched to the incoming surface form (Hankamer (1986)).

The psycholinguistic evidence to date suggests that morphological structure is recognized on-line during word recognition. However, the *number* of morphemes in a word does not directly increase processing time. Suffixes may have separate orthographic representations accessed during reading, but there is no evidence to support separate lexical entries for suffixes that are accessed during auditory word recognition (see Emmorey (1987)). Furthermore, different possible parses may be conducted simultaneously — the evidence that pseudoaffixed words take longer to process is very weak. To investigate effects of false parsing, it would be useful to

look for morphological garden paths. Many hypotheses have arisen within models of syntactic parsing based on garden path sentences (e.g., "The boat floated down the river sank") — do we find similar phenomena for morphological parsing?

Several problems confront the psycholinguist in designing experiments that test for effects of morphological structure in non-european languages. As mentioned at the outset, the experiment must test on-line language comprehension to determine how representations are computed, and several linguistic factors must be controlled such as word frequency and possibly uniqueness point (Marslen-Wilson and Tyler (1981)). For languages which do not have resources that allow such controls, experimentation may be difficult. However, as psycholinguists become more interested in conducting cross-linguistic research (see Cutler (1985)), these resources may become available and some of the questions raised here may be answered.

Chapter 7

Parsing Nominal Compounds in Turkish

Jorge Hankamer
University of California, Santa Cruz

The purpose of this paper is to present an outline of a part of a morphological parser which correctly parses nominal compounds in Turkish. Part 7.1 describes the solution to a relatively simple prior problem, that of parsing words formed by affixation. Part 7.2 presents a generative account of nominal compounding. The problem addressed in this paper is to design a parser which yields the same analyses for any well-formed construction as this generative grammar assigns. Part 7.3 outlines the compound parser.

7.1 Affixation

Affixation in Turkish is relatively straightforward from a generative point of view, though there are many questions of detail (for example, regarding the number and nature of the categories that must be distinguished) which can only be tentatively answered at present.

Briefly, an affixation rule is of the form[1]

[1]Since we are interested in parsing, and the method to be developed is a version of bottom-up parsing, I will systematically represent rules of formation in the form A + B → C, which is to be read "a unit of category A concatenated with a unit of category B counts as a unit of category C". Thus a rule of formation corresponds exactly to the step in parsing where a derived stem is identified.

(1) XX + XXYY → YY

where XX, YY are stem category symbols, and XXYY is an affix category symbol. This rule says that an item of category XX combines (by concatenation) with an item of category XXYY to yield an item of category YY. As a concrete example:

(2) N0 + N0V0 → V0

kilit	+	*le*	→	*kilitle*
'lock'[N0]		[N0V0]		'lock'[V0]

The number of stem categories that must be distinguished in a fully explicit morphological description turns out to be rather large. It is for that reason that I have adopted the convention of representing stem categories with two-character names. Each affix category consists of just those affixes that can attach to a stem of a given category and yield a stem of a given category (not necessarily distinct from the first one); hence it is convenient to name affix categories with four-character names representing their input and output stem categories.

An explicit description of word formation for a language, to the extent that it is agglutinative, can be given by defining the stem and affix categories for the language, and saying which of the stem categories count as word categories.

In most cases, the affix categories will be defined by listing. There are, however, some cases of complex affixes, where the affixes are composed of smaller affixes and might be defined by rule (though no such affix-defining rule, to my knowledge, will be recursive). Since affixes are categorized according to what stem category they attach to and what stem category they produce, the affix categories represent possible transitions between stem categories.

The stem categories, on the other hand, will in most cases require definition by recursive rules which refer ultimately to the affix categories and the primitive root categories, the members of which are listed.

The definition of the stem categories begins with a listing of the members of several primitive stem categories. These are the root categories, which include N0 (nouns), V0 (verbs), A0 (adjectives), P0 (postpositions), Q0 (predicates), and several minor categories. The full set of stem categories is then recursively defined by the listing of primitive categories and affix categories.

Finally, it is necessary to specify which stem categories count as word categories. To take a small example, the stem *o* counts as a word, but the stem *on* does not. In the system implemented here, a stem counts as a word only if there is a transition (or a chain of transitions) to a special category named "WW". Let us consider one example:

(3) **onlara** (o-n-lar-a "to them")

o the root; standing alone, this stem would count as a word, so the grammar must provide transitions to the stem category WW.

on *o* + *n* yields a new stem, which is in a different category from *o*, since it cannot stand as a word, and this stem can have the plural affix attached to it, which *o* cannot. *on* cannot stand as a word, so the grammar must not provide a transition to the category WW except through further affixes.

onlar *on* + *lar* yields a new stem, which is in a different category from *on*. This is necessary because *on* may have *lar* attached to it, but *onlar* may not. *onlar* may stand as a word, so transitions to WW without further overt affixation should be provided for.

onlara *onlar* + *a* yields a new stem, which is in a different category from *onlar*. This is necessary because *onlar* may have *a* attached to it, but *onlara* may not. No further affixation is possible, and *onlara* is a word, so there must be a transition to WW from this category.

7.1.1 The Affixation Parser

Given a model of agglutinative morphology like that presented above, it is relatively straightforward to design a parser which takes advantage of the fact that the morphology (i.e. the definitions of the stem and affix categories) is representable as a finite state transition network, where the root categories represent jumps from the initial state and the final state is WW.

The root and affix categories are specified in two files, a lexicon file and an affix file.

The root lexicon consists of a list of roots with associated category marking (together with a gloss and other grammatical and semantic information, which will not concern us here).

The affixes are listed in categories according to what category of stem they attach to and what category of stem they yield.

The parser tests the word for an initial match with a root; when one is found, the category of the root determines the category in which the next match is sought. If the root is an N0, then the next match must be with a member of a category N0YY, i.e. an affix which can attach to a stem of category N0. If a match with such an affix is found, the process is repeated with the derived stem as base: its category determines the class of affixes to be tested for a match with the next part of the word.

If the end of the word is reached and the final stem is in a category which has a transition to category WW, the parse is deemed a success; otherwise it is a failure. In particular, if at any point there is a remaining substring in which no match with an affix in the required category can be found, the parse is abandoned.

The program incorporates the major phonological rules of Turkish. Roots and affixes are listed in the lexicon and affix files in an abstract underlying representation, and matches with the surface forms are mediated by modifications sensitive to the environment. For example, the past affix is listed as *dı*; this form is modified by operations corresponding to the rules of vowel harmony and stop assimilation before the form is matched to a surface string. Thus *aldı* will be analyzed as

al-dı
take-PST

but *altı* will not be analyzed (wrongly) as

al-tı
take-PST

because the affix-initial consonant would not match the stem-final consonant in voicing.

Similarly, even though the dative affix is listed as *ya*, the string *adamya* will not be parsed. The affix matching routine deletes the buffer *y* from the affix before matching, because the stem *adam* ends in a consonant.[2]

Here is an example of the parsing of a moderately complex word. Given the form *gözlükçülerimizden* the analysis proceeds as follows:

First the root *göz* 'eye' is recognized; its lexical entry specifies that it is an N0; then an affix is sought that can attach to N0 and initially matches the string after *göz*. *lg* matches, and is in the category N0N0; its derived category is N0, so *gözlük* counts as an N0. Now the program again seeks a match with affixes that can attach to N0. The affix *cı* is among them, and matches the string *çü* that follows *gözlük*. The category of the new affix is N0N0, so the next affix again has to be in a category N0YY. This time no overt affix is found to match the next part of the word, so a

[2]The system as currently implemented has some imperfections. It does not, for example, have a way of dealing adequately with morphophonemic alternations that are conditioned by the number of preceding syllables, as is the case with the causative and aorist affixes. This causes the parser to accept some false parses (*al-t-tı*[take-CAUS-PST] for *altıtı*, for example). It also has no way of knowing that *gideme* [go-ABIL-NEG] is not a likely imperative, but that is probably a semantic rather than a morphological limitation.

jump is taken to category N1.[3]

From N1 there are jumps to N2 and to N3. If the jump to N2 is taken, the next affix must be the plural affix *lar*; if the jump to N3 is taken, the next affix must not be the plural. Both paths will be attempted, but the one which commences with a jump to N3 will fail, since there is no way to reach the end of the word along that path. The path commencing with a jump to N2 leads to the correct analysis.

The only affix that can attach to an N2 is the plural affix *lar*, which is the sole occupant of category N2N3; this matches the *ler* after *gözlükçü*, so *gözlükçüler* is analyzed as a stem of category N3. Note that a stem of category N3 may or may not have the plural affix.

From N3 there are jumps to N4 (from which there is further progress only by finding a possessive affix), to N7 (from which there is no escape except by finding a case affix), and to N9 (no further affixation required). To find an analysis for the given word, the parser must recognize the possessive affix *ımız* (category N4N5) as matching *imiz*, so that *gözlükçülerimiz* is analyzed as a stem of category N5; then a jump is taken to N7, from which the ablative affix *dan* (category N7N9) can be matched to yield *gözlükçülerimizden* as a stem of category N9. There is a jump from N9 to NN and another jump from NN to WW, so the analysis is a success.

I will not discuss in detail the reasons for all of the category distinctions. Those which figure in the operation of the compound parser outlined in section 7.3 are N0, the category of noun roots and noun stems derived by derivational affixes; N3, the category consisting of noun stems with or without the plural affix but lacking possessive or case affixes; N4, the stem category which must acquire a possessive affix; and N6, which is an N4 with the third person singular possessive affix (affix category N4N6) attached to it.

The parser described in this section is extremely simple, yet it seems to be adequate to deal with the agglutinative aspects of Turkish morphology.[4] What it cannot do, without some sort of radical embellishment, is assign correct analyses to compounds.

7.2 The Structure of Nominal Compounds

The nominal compounds to be treated here are those of the "indefinite izafet" construction (Lewis (1956), pp. 41–43). As a first approximation, their structure might

[3]Category N1 is distinguished from category N0 because adjectival stems (in particular participles derived from verbs) can function as noun stems, receiving the plural, possessive, and case affixes that affix to noun stems, but cannot receive the N0YY derivational affixes.

[4]After this paper was substantially written, I learned that a morphological parser based on similar principles has been developed for Finnish by Koskenniemi (1983).

be described as

(4) N3 + N6 → N0

where N3 is a noun stem with or without a plural affix[5] and N6 is a noun stem with the third person singular possessive affix. This affix is the sole member of the category N4N6; it is glossed POSS in the examples below.

(5) a. *çocuk kitab-ı*
child book-POSS
'children's book'

b. *ev kapı-sı*
house door-POSS
'house door'

c. *kız okul-u*
girl school-POSS
'girls' school'

This construction must be distinguished from the "definite izafet" construction, exemplified in (6):

(6) a. *çocuğ-un kıtab-ı*
child-GEN book-POSS
'the child's book'

b. *ev-in kapı-sı*
house-GEN door-POSS
the door of the house'

c. *kız-ın okul-u*
girl-GEN school-POSS
'the girl's school'

In the definite izafet construction, the first member is in the genitive case. The definite izafet construction is in fact quite different from nominal compounding. Its structure is represented by the following rule:

(7) NP[gen] + NP[poss] → NP

[5] Lewis (1956) (p. 44, footnote 1) indicates that he thinks the first members of compounds should not have the plural affix. This is probably incorrect, however, for I have seen such things as *körler okulu* 'school for the blind', and forms like *uluslararası* 'international' appear to be well established.

Here "NP" is the phrase category Noun Phrase, not a morphological category. NP[gen] is a NP whose head noun has a genitive affix on it, and NP[poss] is an NP whose head noun has a possessive affix on it.[6]

This claim is supported by the fact that in the "definite izafet" either the first or the second member may be modified by any of the kinds of complements that are found in NP's:

(8) *küçük çocuğun kitabı*
'the small child's book'

çocuğun küçük kitabı
'the child's small book'

küçük çocuğun küçük kitabı
'the small child's small book'

çocuğun bir kitabı
'a book of the child's'

adamın çocuğunun kitabı
'the man's child's book'

senin gördüğün çocuğunun kitabı
'the book belonging to the child that you saw'

çocuğunun senin gördüğün kitabı
'the book that you saw that belongs to the child'

while in the "indefinite izafet" all modifiers must precede the first member of the construction and semantically apply to the construction as a whole:

(9) *küçük çocuk kitabı*
'small children's book' (small book for children)

**çocuk küçük kitabı*

bir çocuk kitabı
'a children's book'

**çocuk bir kitabı*

senin ğordüğün çocuk kitabı
'the children's book that you saw'

**çocuk senin gördüğün kitabı*

[6]The possessive affix in this case is any N4YY affix, not necessarily N4N6; it must agree with the genitive affix in NP[gen].

There is one further construction which must be set aside. Turkish grammarians (cf. Lewis (1956), p.42) regarded constructions like those in (10) as compounds:

(10) *tahta kutu* 'wood box'
demir perde 'iron curtain'
bakır tas 'copper bowl'

In this construction the first member must be a noun denoting a material, the second a noun denoting a type of object, and the whole denotes an object of that type made from that material. Neither member is marked by a suffix.

I deal with these constructions by entering the material nouns twice in the lexicon, once as nouns and once as adjectives. This accounts for the fact that such material nouns, when combined with nominal compounds of the indefinite izafet type, invariably precede the first member of the compound, and cannot intervene between the two members:

(11) *tahta sigara kutusu* 'wooden cigarette box'
**sigara tahta kutusu*

Henceforth, then, the term "nominal compound" will mean an instance of the "indefinite izafet" construction.

If rule (4) describes the basic structure of a nominal compound, it is necessary to formulate some transformational rules to modify that structure under certain conditions. The remainder of this section is devoted to the development of a transformational description of nominal compounds proceeding from this assumption. In the following section I will show how to describe the structure of compounds without such transformations.

One reason there must be some transformational modification of compounds if (4) is assumed to be the basic rule of formation is that any time two possessive morphemes (of any person and number) would, according to the rules of combination, appear adjacent to each other, the innermost (leftmost) one is deleted. Thus if the nominal compound is made the right member of a definite izafet construction, we find the following:

(12) *Hasan-ın ev anahtar-ı* 'Hasan's house key'
**Hasan-ın ev anahtar-ı-sı*
Hasan-ın at araba-sı 'Hasan's horse cart'
**Hasan-ın at araba-sı-sı*

The possessive morpheme never appears immediately preceding another possessive morpheme. When the rules of combination would produce such a sequence, the leftmost of the two adjacent possessive morphemes vanishes:

(13) N4N6 → Ø/ ___ N4YY (N4YY is the class of possessive affixes)

We know it is the leftmost possessive morpheme that vanishes because In genitive-possessive constructions the two may represent different persons:

(14) *at araba-sı* 'horse cart'
(benim) at araba-m 'my horse cart'
**at araba-sı-m*

The 3sg possessive morpheme {(*s*)*ı*} vanishes when adjacent to a following 1sg possessive morpheme.

This same rule applies no matter what the source of the possessive morphemes. In particular, the nominal compounding rule is recursive, and indeed we find compounds in which one member or the other is itself a compound; see Figure 7.1.

türk dil-i gramer-i 'turkish language grammar'
türk dil kurum-u 'turkish language society'

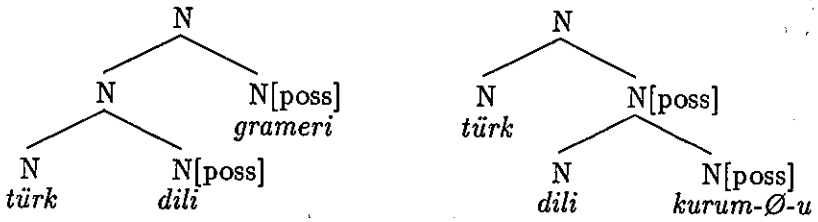


Figure 7.1: Recursive compounding

Note that when it is the second member that is a compound, we find not two possessive morphemes but one.

Similarly, the possessive morpheme vanishes when adjacent to a following derivational affix (Lewis (1956), p. 50):

(15) *su yol-u* 'water conduit'
su yol-cu 'person responsible for the upkeep of water conduits'

(16) N4N6 → Ø/ ___ N0YY

These examples are interesting, because they show not only that the domain of poss-deletion is more general, but also that compounds may undergo derivational affixation. In the terms of the model of morphological structure sketched in section 7.1, this means that a compound can count as an N0.

Another transformational rule affecting the appearance of nominal compounds is one which determines the order of plural and possessive suffixes. The relative order of plural and possessive morphemes is fixed: no matter what the origin, a plural morpheme must precede an adjacent possessive morpheme. Morphemes in the order possessive-plural can arise under the current assumptions because rule (4) makes a compound an N0, and N0 (after jumps to N1 and N2) can receive a plural affix; see Figure 7.2.

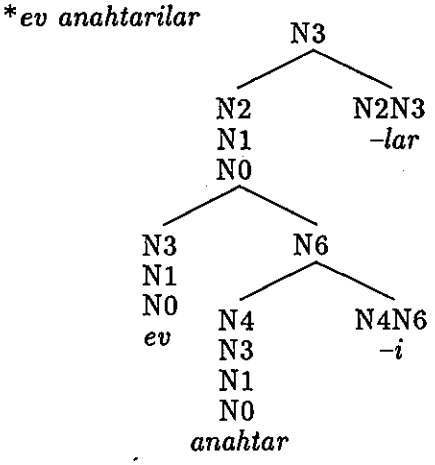


Figure 7.2: Structure of **ev anahtarilar*

Thus we will need a metathesis rule:

(17) N4N6 N2N3 → N2N3 N4N6

Thus we have, as the plural of *ev anahtarı* "house key"

(18) *ev anahtarları* 'house keys'
**ev anahtarılar*

There is yet one more transformation to be formulated. The grammar now generates sequences of two plural affixes. This is because the stem *anahtar* is eligible to receive

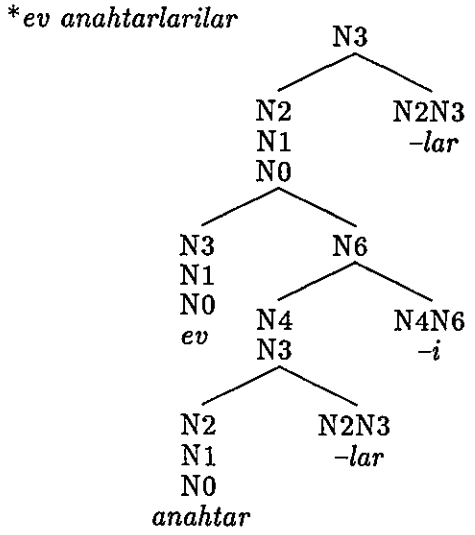


Figure 7.3: Structure of **ev anahtarlarilar*

a plural affix just as the whole compound is, in which case the structure would be as in Figure 7.3.

Once rule (17) permutes the possessive affix with the outer plural affix, the two plural affixes are adjacent. Since such sequences never appear superficially, there will have to be a transformation deleting one of two adjacent plural affixes.

(19) N2N3 → Ø/ ___ N2N3

The construct *ev anahtarları* is in fact several ways ambiguous, because of the conflation of the forms corresponding to "house keys", "his/her/its house keys", "their house keys", "their house key".

Rule (17) may feed both the possessive deletion rule (13) and the plural deletion rule (19).

Together with the affixation rules described in section 7.1, rules (4), (13), (16), (17), and (19) account for the structure of nominal compounds. This description is a formal version of the informal description given by Lewis (1956).

Some examples are shown in Figures 7.4, 7.5, and 7.6.

In Figure 7.4, rule (17) permutes the inner poss (N4N6) morpheme with the plural (N2N3) morpheme; rule (13) then deletes the inner poss morpheme, and the resulting string is shown in Figure 7.4. The rules I have written are not explicit

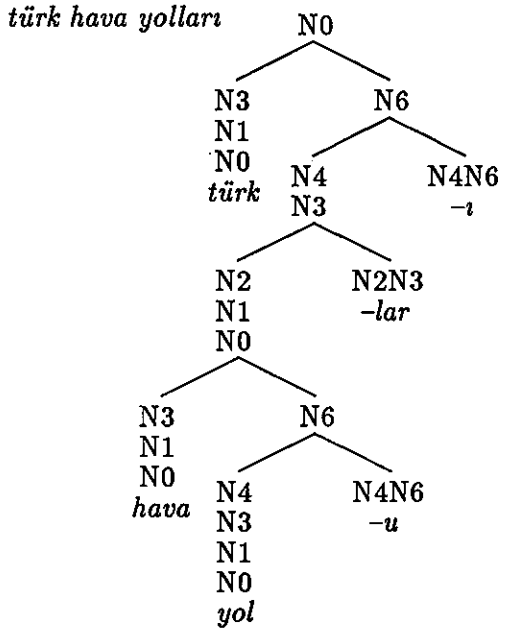


Figure 7.4: Structure of *türk hava yolları*

about certain details of the intermediate structures, in particular what the structural effects of the permutation rule are; this is because those details do not appear to be empirically determinable, and as will be shown in the next section are irrelevant to the parsing analysis.

In Figure 7.5, nothing happens but the deletion of the poss morpheme by rule (16).

In Figure 7.6, rule (13) deletes the innermost poss and rule (16) deletes the next one, assuming a cyclic application of these rules. It doesn't matter whether this is assumed or not, since any other order of application will have the same net effect.

The description sketched in this section is transformational, in that it assumes that the construction rules produce basic structures which are then modified by permutation and deletion. While this model describes the desired forms in a generative fashion, it is not immediately amenable to a practical parsing interpretation. It is, I believe, possible to construct a parser corresponding to this grammar, i.e. one which accepts the same strings and assigns them the same underlying structures. Such a parser, however, would have to undo much of the work of the affixation parser in unravelling the word structure and replacing the missing N4N6 morphemes.

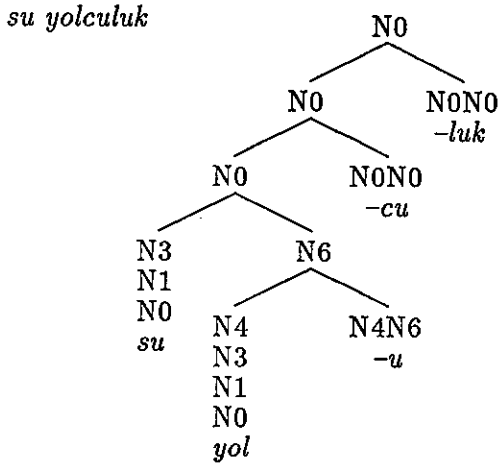


Figure 7.5: Structure of *su yolculuk*

This is not just a mechanical inconvenience. There is something fundamentally wrong with this analysis, whether we want a parser or not. The basic word formation rules, as represented in the affix transition network, stipulate that a word may have at most one possessive affix and at most one plural affix, that no possessive affix precedes a derivational affix, and that if there are both a plural and a possessive affix, the possessive must precede the plural. The transformational rules (13), (16), (17), and (19) have the effect of restoring the second word in a compound to the state that the basic rules say it must have, in effect undoing the work of rule (4).

We must suspect that something is wrong, and in particular that rule (4) is wrong. Note that similar observations formed the basis for Chomsky's (1969) attack on the transformational analysis of derived nominals, for the situation is the same: the transformational account is suspect because the transformations are required to derive structures which are already described by basic rules.

In the next section, a non-transformational analysis will be outlined which accepts exactly the same strings as are generated by these rules, while assigning them correct structures. The analysis will be presented as a parser, but (like the analysis of agglutination outlined in section 7.1) it can equally well be regarded as a generator.

türk su yolcu dergisi

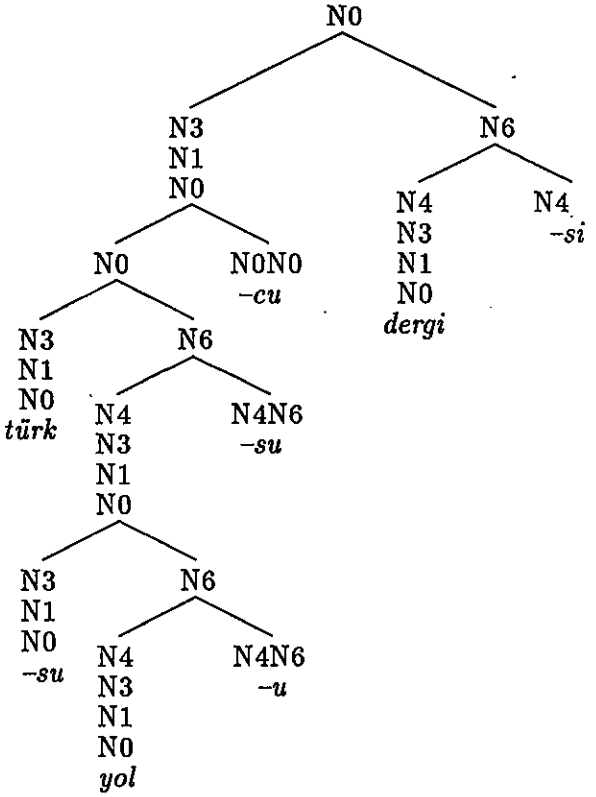


Figure 7.6: Structure of *türk su yolcu dergisi*

7.3 Parsing Nominal Compounds

The generative grammar developed in section 7.2 admits the correct class of nominal compounds.

The problem now is to devise a parser which will accept as input a string of words and determine whether the string constitutes a well-formed nominal compound or not, assigning it a correct analysis if it does.

The course adopted here is to allow the affixation parser described in section 7.1 to provide parses for each word, and then to pass the resulting information to a secondary parser which assigns an appropriate structure to the compound, or rejects it if no structure can be assigned. The structures assigned will differ in certain details from those assigned by the grammar of section 7.2, but the major

structural relations (the relations between the nominal members) are the same.

The secondary parser proceeds by identifying possible first and second members of compounds in the bracketed string output by the suffixation parser. It then assigns new bracketing by an "annexation" operation, which makes the left member a sister of the right member, leaving other structural relations intact. The creation of a compound may produce new possibilities for combination, since compounds themselves may be candidates for membership in larger compounds.

The first task, then, is to identify possible left and right members of simple compounds. The grammar of section 7.2 says that compounds are composed of [N3 + N6]; but the transformations (13) and (16) may strip the N4N6 suffix off, so that the affixation parser cannot analyze the form as an N6. If we look at the output of the affixation parser as applied to a compound, we find that simple compounds may be identified as [N3 + N0] followed immediately by a derivational affix or by a possessive affix with at most a plural affix intervening.

This suggests that we may begin by identifying the sequence [N3 + N0] as a compound if the environment immediately following is one of the two mentioned above. Once simple compounds are identified, we may observe that an N3 may combine with a following compound to form a compound of the *türk dil kurumu* type; and that a compound plus its possessive affix (with an optional plural affix in between) can combine with a following right member to form a compound of the *türk dili grameri* type.

All this is summarized in the following two rules:

$$(20)\quad \text{N3 N0} \rightarrow \text{N0} \Big/ \text{\_\_} \left\{ \begin{array}{l} \text{N0N0} \\ \text{(N2N3) N4YY} \end{array} \right\}$$

$$(21)\quad \text{N3 N0 (N2N3) N4N6} \rightarrow \text{N3}$$

The first rule identifies a sequence N3 N0 as an N0 if it is in one of the contexts where compounds can occur; the second rule says that a compound plus its plural (if present) and 3sg possessive affixes counts as an N3 for purposes of further compounding. These rules are strictly cyclic.

The rules are cyclic because units created by them may be components of larger units also created by them. They are "strictly" cyclic because it would not do to allow either rule to iterate on the same domain, yielding multiple analyses like the following:

(22) *su yolcu*

$$[su]_{N}\ [[yol]_{N}\ -cu]_{N}$$
$$3 \qquad 0 \qquad 0$$

$$[[[su]_{N}\ [yol]_{N}\ ]_{N}\ -cu]_{N}$$
$$3 \qquad 0\ 0 \qquad 0$$

$$[[[[su]_{N}\ [yol]_{N}\ ]_{N}\ ]_{N}\ -cu]_{N}$$
$$3 \qquad 0\ 0\ 0 \qquad 0$$

$$[[[[[su]_{N}\ [yol]_{N}\ ]_{N}\ ]_{N}\ ]_{N}\ -cu]_{N}$$
$$3 \qquad 0\ 0\ 0\ 0 \qquad 0$$

The operation represented by rule (20) may be regarded as chomsky-adjunction of the left member to the right member.

Here is an example:

(23) *türk dili grameri*

The affixation parser produces the following output:[7]

türk	*dil*	*-i*	*gramer*	*-i*
N	N	N	N	N
3	0	6	0	6

Rule (20) annexes *türk* to *dil*:

türk	*dil*	*-i*	*gramer*	*-i*
N	N	N	N	N
3	0	6	0	6
	N			
	0			

Rule (21) says *türk dili* counts as an N3:

[7] I am omitting null transitions which are ignored by the secondary parser.

türk	*dil*	*-i*	*gramer*	*-i*
N	N	N	N	N
3	0	6	0	6
	N			
	0			
		N		
		3		

Rule (20) combines *türk dili* with *gramer*:

türk	*dil*	*-i*	*gramer*	*-i*
N	N	N	N	N
3	0	6	0	6
	N			
	0			
		N		
		3		
			N	
			0	

Rule (21) now says that the whole expression counts as an N3.

türk	*dil*	*-i*	*gramer*	*-i*
N	N	N	N	N
3	0	6	0	6
	N			
	0			
		N		
		3		
			N	
			0	
				N
				3

The effect of these reanalyses is to extend the categories N0 and N3. But because the compounding analysis is a second stage operation, presuming that affixation has already been dealt with by the affixation parser, there is no possibility of the system accepting a possessive morpheme affixed to one of these secondarily derived N3's. Sequences of adjacent possessive morphemes are ruled out once and for all by the affixation parser.

The analysis of an example like *türk dil kurumu* will proceed as follows:

türk	*dil*	*kurum*	*-u*
N	N	N	N
3	3	0	6
	N		
	0		
		N	
		0	
			N
			3

Finally, the analysis of *su yolcu dergisi*:

su	*yol*	*-cu*	*dergi*	*-si*
N	N	N	N	N
3	0	3	0	6
	N			
	0			
		N		
		3		
			N	
			0	
				N
				3

The structures assigned by these rules correspond in all major respects to those assigned by the generative rules of section 7.2. The structural relations assigned in the *su yolcu* type are clearly correct, as shown in Figure 7.7.

su yolcu

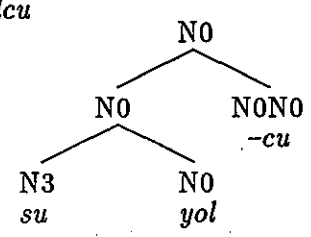


Figure 7.7: Structure of *su yolcu*

It may seem bizarre at first glance that the possessive suffix in *su yolu* is not treated as part of the compound but rather as a sister to it; see Figure 7.8.

This appears to be empirically harmless, however. The structure assigned by

su yolu

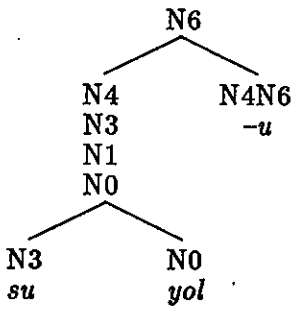


Figure 7.8: Structure of *su yolu*

the affixation parser already says that the possessive morpheme is part of the last word in the compound, and that fact is not changed. The compounding parser may be regarded as concerned only with assigning structural relations among the N0s in the compound, so it treats the possessive morpheme as if it were as high up as possible just to have it out of the way.

I will conclude this section with a discussion of one interesting example which demonstrates a slightly more complex interaction of the compounding rules. The following example is, in the first place, ambiguous:

(24) *türk yolculuk dergisi*

It can be analyzed several ways, depending on the order in which the elementary compound parts are combined:

	türk	*yol*	*-cu*	*-luk*	*dergi*	*-si*
(25)	N	N	N	N	N	N
	3	0	0	3	0	6

One possible initial combination, that of *yolculuk* with *dergi*, leads to a compound of the same structure as *türk dil kurumu*, discussed above. It would mean 'travel journal of Turkey'.

If instead *türk* is first combined with *yolcu*, we get the following analysis:

(26) 'journal of Turkish travel'

türk	*yol*	*-cu*	*-luk*	*dergi*	*-si*	
N	N	N	N	N	N	
3	0	0	3	0	6	
		N				Rule (20)
		0				
			N			
			3			
				N		Rule (20)
				0		
					N	Rule (21)
					3	

Note that the application of Rule (20) that makes *türk yolcu* an N0 also makes *türk yolculuk* an N0, and of course also an N3.

If *türk* is first combined with *yol*, we get the following analysis:

(27) 'journal of Turkish road work'

türk	*yol*	*-cu*	*-luk*	*dergi*	*-si*	
N	N	N	N	N	N	
3	0	0	3	0	6	
	N					Rule (20)
	0					
		N				
		0				
			N			
			3			
				N		Rule (20)
				0		
					N	Rule (21)
					3	

Here the first application of rule (20) has made *türk yol* an N0, which has the consequence that *türk yolcu* and *türk yolculuk* are also N0. *türk yolculuk* is also an N3.

I have presented the analysis as a two-stage parser. It can equally well be viewed as a two-stage generator. The affixation rules can be viewed as generative rules which build stems of various types: N0, N3, N6, etc. The two rules (20) and (21) can be viewed as generative rules which, like the generalized transformations of Syntactic Structures, put the prefabricated stems together in specified ways. Under this view, rule (20) is an embedding transformation.

7.4 Conclusion

The algorithm outlined in section 7.3 recognizes exactly the compound nouns generated by the rules presented in section 7.2. The structures assigned to these compounds differ from those assigned by the rules of section 7.2 only in the question of the structural relation between the possessive morpheme and the rest of the compound.

The virtue of this analysis, besides the fact that it yields a correct and rather straightforward parser, is that the restrictions on affix occurrence and order, which are properties of simple words, are stated once and for all in a component of the grammar which defines simple word structure. None of the patch-up transformations which are required under the analysis sketched in section 7.2 are needed under this analysis.

On the other hand, it might be regarded as a virtue of the transformational analysis that it provides multiple derivations for ambiguous expressions such as *ev anahtarları*. The treatment of section 7.3 gives this expression only one analysis. If a correct interpretation of the expression is to be provided as well as a correct parse, it will be necessary to provide for some rules of semantic interpretation which specify the possible contributions of the possessive and plural morphemes to the interpretation of the expression.

Chapter 8

Parsing Quechua Morphology For Syntactic Analysis

Karen Wallace

8.1 Introduction

This paper explores some of the questions and problems encountered in designing and implementing a morphological parser for a language with complex word structure. In the approach taken here, the main goal of the morphological parser is to provide the syntactic parser with the information it needs about the words in the sentence. My goal in this project was to construct a parser which embodies a linguistically satisfying analysis.

This parser was implemented for Bolivian Quechua. The formal morphological apparatus of Quechua is just like that of Turkish: it is exclusively suffixing in nature. Three basic classes of suffixes can be distinguished: (1) derivational suffixes, (2) inflectional suffixes, and (3) phrasal clitics.

Derivational suffixes build stems; the features which are affected are not "relevant to the syntax" in the particular sense outlined in Anderson (1982). A feature which simply adds or absorbs an argument position in the subcategorization frame of a verb is not syntactically relevant in this sense, because it only affects the lexical insertion possibilities of the verb—it is not crucial to or dependent on any syntactic operations beyond lexical insertion. These suffixes are in general quite productive in this language, but the composition of parts often yields stems whose meanings are not predictable.

Inflectional features, on the other hand, either depend on syntactic structure or must be accessible to some syntactic operation. For example, the case that a noun bears depends on the syntactic configuration it occurs in; and a noun's inherent features of person and number must be accessible to rules of agreement.

Phrasal clitics can be distinguished from inflectional suffixes by their ability to attach to virtually any category. Their content is generally semantic, rather than grammatical, and they never change the category of the thing they attach to.

Quechua dialects can be divided into two mutually unintelligible groups: the Quechua I/B dialects spoken in Central Peru, and the Quechua II/A dialects spoken in Ecuador, Bolivia, and southern Peru (Parker (1969a)). The dialect of Quechua under consideration here is spoken in Cochabamba, Bolivia (Herrero and Sánchez de Lozada (1978), Bills et al. (1969), Lastra (1968)); structurally it is quite similar to the dialect of Cuzco, Peru. The morphological system is complex enough for us to be able to assume with confidence that words must be parsed and not merely listed in the lexicon. Quechua has a very regular inflectional system and a rich derivational morphology; there are a number of interesting phenomena, such as portmanteau suffixes, an "empty morph", and discontinuous dependencies. These and other aspects of the morphological system will be discussed in turn, with particular attention to how they are handled in computer parsing.

8.1.1 Background in Morphological Parsing

Morphological parsing has a fairly short history. The two morphological parsers I am familiar with are KIMMO (Koskenniemi (1984), Karttunen (1983)) and *keçi* (Hankamer (1986)). Both encode the morphotactic system of the language with a finite state machine. In parsing a word, the root is found by matching the leftmost part of the word against a (possibly very large) lexicon of roots. A successful match establishes the initial state; the category associated with the root and the rest of the word are then passed through the finite state morphotactic network by matching affixes and their category transitions or continuation classes. The word is successfully parsed if the machine ends up in a designated final state and has no more of the word left to parse. The output of a successful parse is a collection of the labels associated with the arcs traversed.

Phonological alternations are handled differently in the two parsers. In *keçi*, a candidate form is altered by applying phonological rule functions to it prior to actual comparison of forms. In KIMMO, surface and underlying forms are on two tapes; pairs of corresponding segments are passed through finite state automata which encode the phonological alternations (see Kay (1983), Koskenniemi (1984)).

It was an attempt to implement these parsers for Quechua morphology that

motivated a different approach. In fact, this parser adopts principles of *keçi* with a few crucial differences.

- In the *KIMMO* and *keçi* systems, the morphotactics are encoded into a finite state network by specifying a state transition or continuation classes for each morpheme. However, the fact that there are discontinuous dependencies in morphology (and in particular, in Quechua morphology) make a finite state network an unsavory prospect. In a finite state system there is no way to relate elements occurring on non-adjacent arcs: any element A whose occurrence or interpretation depends on the occurrence of a non-adjacent element B must be given a dual incarnation—one as it occurs with B and one as it occurs without B.

- No distinction between derivational and inflectional morphology is made in these systems, in that the output of a successful parse is a homogeneous collection of the information associated with each morpheme recognized. Such a collection gives both too much and not enough information if the goal of the morphological parser is to provide the syntax with only the morphosyntactic features of the word (see Anderson (1987)). What is needed is a system which (a) parses derivational morphology and puts out a lexical representation for the complex stem; and (b) parses inflectional morphology and puts out a morphosyntactic feature matrix.

8.1.2 How Qpop works

In an attempt to address these issues, "Qpop" was implemented in Pop-11 using a transition network augmented with registers holding the syntactic information gathered at each stage of the parse. In the sections that follow I hope to show that something more than a finite state morphotactics is appropriate for Quechua, and that the division made here between derivational and inflectional morphology is a correct one. However, the basic design of the parser follows that described in Hankamer (1986).

In Qpop, the input word is matched against a lexicon of stems; the matching procedure includes a call to the procedure which applies phonological rules. This gives a list of the possible lexical stems.

For each item in the list of possible lexical stems, a recursive procedure is called to do the following: the remainder of the word following the matched stem is matched against the list of derivational suffixes with the same category. Part of this procedure is to apply phonological rules conditioned by particular morphological elements, such as the *u-a* rule discussed in section 8.2.2. For each successful match, a new stem is

built with an amended representation and added to the list of possible stems. If no suffixes match, the set of possible stems is simply the set of lexical stems.

For each item in the list of possible (lexical and derived) stems, the category, feature registers, and form are matched against the set of inflectional suffixes. For each successful match, the feature registers are affected accordingly. A morphosyntactic representation (MSR) is built from whatever is in the feature registers when the end of the word and the end-of-word category are encountered simultaneously. Default values of features are inserted when the MSR is built.

The output of the parser is a set of stems (corresponding to the output of the derivational component) and a list of MSRs for each. Stems for which no MSRs were built are failed parses.

The rest of this paper is organized as follows. Section 8.2 briefly describes phonological alternations in Bolivian Quechua and the way they are handled here. Section 8.3 discusses the derivational morphology, section 8.4 describes aspects of the inflectional morphology, and section 8.5 concludes.

8.2 Phonology

Before the influence of Spanish, Quechua had a three-vowel system; *e* and *o* were predictable variants of *i* and *u*, occurring adjacent to a uvular stop. Bilingual speakers now have the five-vowel system of Spanish (Herrero et al. (1978), Albó (1969)); hence, mid vowels are treated here as underlying unless they are seen to alternate with high vowels, despite the fact that in native words it is rare to find a mid vowel which is not adjacent to a uvular consonant.

Voiced stops occur only in Spanish loan words, and in native words stops occur only syllable-initially. There is a voiced retroflex spirant which is an allophone of the flap *r* occurring word-initially. It is contrastive, however, in Spanish loan words like *wurru* 'donkey', and will be represented with the symbol *rr* in those words.

Oral stops do not surface syllable-finally. Bills et al. (1969) suggest that the velar and uvular fricatives which appear syllable-finally are allophones of the corresponding stops. This is plausible since there is no velar–uvular fricative contrast syllable-initially.

In some Bolivian dialects (see Parker (1969b) and Albó (1969) for discussion) the distinction between velar and uvular fricatives has been neutralized, and *j* is written to represent the back fricative. In dialects which retain the contrast, the syllable-final velar or uvular fricative is represented as a non-contrasting variant of the velar or uvular stop, and *j* occurs only syllable-initially.

The syllable canon is (C)V(C) word-initially and CV(C) elsewhere. While word

stems are often vowel-initial, all native suffixes have an initial consonant, and many have an initial CC sequence.

Phonological rules are compiled as procedures operating on a candidate form prior to matching, with access to both the surface word and the current candidate, as in Hankamer (1986). Features such as [+high] are defined with set membership; the conditions on the application of rules are based on the feature properties of the segments involved and their surface environments. There is no internal featural representation of segments, so changes are actually brought about by replacing entire segments. I will eventually implement a version which incorporates feature representations and feature changes to reflect phonological processes more accurately.

8.2.1 Phonological Lowering

The only pervasive and regular phonological rule that I know of in this dialect of Quechua is vowel lowering in the environment of a uvular stop.

(1) Lowering

$$[+\text{syll}] \longrightarrow [-\text{high}] \;/\; \underline{\quad}\; ([+\text{cons}, +\text{son}])\ [+\text{cons}, +\text{back}, -\text{high}]$$
$$[+\text{cons}, +\text{back}, -\text{high}]\ \underline{\quad}$$

The features [+back, –high] pick out the class of uvular stops. Notice that the assimilation operates left and right, yet this is not a mirror image rule since no intruding sonorant is possible when the affected vowel is to the right of the uvular. Here are a few examples showing the operation of this rule.

(2) ri–rqa → rerqa[1]
go–past
'she/he went'

pisi–qa → piseqa
little–topic
'a little (topic)'

perqa–ita → perqeta
wall–dimin
'little wall'

pi–qpata → peqpata
who–gen
'whose'

8.2.2 The u–a Rule

The derivational "modal" verb suffixes *yku, rqo, ku* are realized as *yka, rqa, ka* if they immediately precede *mu* or *pu* (also derivational verb suffixes) or *ka* (the form of *ku* resulting from the same rule). An instance of *ku* which is not affected by the rule does not trigger it either; compare *apa-rqa-ka-mu-y* with *apa-rqo-ku-y*. Nor is it the case that *rqo* is directly affected by *mu* or *pu* despite *ku* intervening, since the intervention of any other suffix blocks the rule, as shown by words like *apa-rqo-ri-mu-y*.

This alternation does not occur in roots; the *ku* of *riku-* 'see' is unaffected. Nor is it triggered by other, homophonic sequences such as *pu* in the clitic *puni* 'certainly'.

This is an alternation which does not appear to be triggered purely by either phonological form or morphological features, but both. If we treat this as a purely phonological alternation, reference to the form of the triggering elements must be explicit enough to pick out *mu* and *pu* and *ka* but no other suffixes or parts of suffixes. This can be done, but it seems wrong to rely solely on phonological shape when it is clear that morphological information is relevant to the operation of the rule. It seems that we are forced to list the different forms of the affected suffixes in the lexicon together with a definition of the set of suffixes triggering their occurrence. Because the class of suffixes which trigger and/or undergo the alternation is small and has no distinctive morphological or phonological characteristic (for example, only the *ka* form of the reflexive suffix is a trigger), it seems that listing the forms in the lexicon is an accurate way to analyze it. During parsing, recognizing the *a* form of an affected suffix results in the requirement that the next suffix recognized belongs to the set of triggers.

8.2.3 Diminuitives

In Bolivian Quechua, the native diminuitive suffix has been abandoned in favor of the Spanish diminuitive suffix; however, the distribution of allomorphs in Quechua is interesting. Compare the following forms (Herrero et al. (1978)):

[1]The orthography used here follows one of the systems of orthography I have found in Bolivian texts. It is more or less phonemic, and standard except for the following: *ch* and *sh* represent the palato-alveolar affricate and fricative, respectively; an apostrophe following a stop or affricate indicates an ejective, an *h* following a stop or affricate indicates aspiration, and *ll* is a lateral glide (distinct from the palatal glide *y*).

(3) Quechua Diminuitive Forms

t'ika	'flower'	perqa	'wall'
t'ikita	'little flower'	perqeta	'little wall'
runtu	'egg'	alqo	'dog'
runtitu	'little egg'	alqetu	'little dog'
misi	'cat'	wawqe	'brother (of male)'
misisitu	'little cat'	wawqesitu	'little brother'
yan	'road'		
yansitu	'little road'		

The top four pairs show that in noun stems ending in a back vowel, the final vowel is deleted and *itu* or *ita* is suffixed, depending on whether the stem vowel was *u/o* or *a*. The second three pairs show that noun stems ending in a front vowel or a consonant simply suffix *situ*, with no *u/o–a* alternation or vowel deletion.

The interesting problem here is in recognizing the lexical stem of a word like *alqetu*, since logically this must be done prior to the recognition of the diminuitive suffix. In the parser, just prior to attempting to match stems, a procedure checks for sequences of *i/e–t–u/a* at the appropriate place in the surface form according to the length of the current stem candidate. If it finds such a sequence (and if the stem vowel agrees in height with the final vowel of the diminuitive sequence), the last vowel of the candidate is merely truncated so that it will be recognized as a possible stem.

A better approach (as yet unexplored) might be to blindly allow any stem ending in a back vowel to be recognized with either its underlying final vowel or a front vowel. In the second case, a flag would go up to ensure that the proper diminuitive suffix is recognized.

8.3 Derivational Morphology

8.3.1 "Modal" Suffixes In Verbs

Verbs in Quechua have roughly the following morphological structure:

stem — modal suffixes — inflectional suffixes

Verb stems in Quechua can be derived from other verb stems by the addition of so-called "modal suffixes." An incomplete list of these suffixes is given below, with

rough glosses to indicate the semantic and/or subcategorization changes effected in the verb.

(4) Some Modal Suffixes in Cochabamba Quechua

- *ysi* "help [someone] do X", e.g. *awa-ysi-yki* "I help you weave"
- *naya* "feel like doing X", e.g. *puñu-naya-ni* "I feel like sleeping"
- *ykacha* "do X here and there", e.g. *puri-ykacha-sa-n* "He's walking around"
- *yku* "finality/decisiveness", e.g. *ranti-yku-saq* "I'm really going to buy [it]"
- *rqo* "honorific", e.g. *mikho-rqo-y* "Please eat! (very polite)"
- *rpa* "do X suddenly", e.g. *saya-rpa-ri-n* "He suddenly got up"
- *ri* "polite", e.g. *jaywa-ri-mu-wa-y* "Please hand [it] to me"
- *chi* "cause to do X", e.g. *mikhu-chi-ni* "I cause [someone] to eat (i.e., feed [someone])"
- *na* "reciprocal", e.g. *qhawa-na-ku-ncheq* "We watch each other"
- *ku* "reflexive", e.g. *chura-ku-ni* "I put [clothes] on myself (I got dressed)"
- *mu* "directional", *apa-mu-ni* "I bring [it]" (cf. *apa-ni* "I take [it])"
- *pu* "benefactive", *apa-m-po-rqa-yki* "I brought [it] for you"

According to Herrero and Sánchez de Lozada (1978), it is common in conversation for a verb to have three or four of these suffixes, but it is rare for there to be more than six. Many scholars (Yokoyama (1951), Lastra (1968)) ascribe a fixed order to them and then enumerate exceptions to this order. Herrero and Sánchez de Lozada (1978) say that the order is fixed in general, but that the ordering is variable in some instances. In fact, from the examples given there, it seems best to treat most of these suffixes as unordered.

Muysken (1977) discusses the properties of these suffixes in various dialects of Quechua in an attempt to establish whether they should be treated (1) as higher verbs, with a raising or clause union transformation to derive the surface forms; (2) as semantic features of the verb; or (3) as unanalyzed parts of stems listed in the lexicon, with redundancy rules to account for the regularities. He then shows that neither of the first two analyses is adequate alone. A pure "higher verb" analysis predicts that variable orderings will always reflect differences in semantic scope; this fails because some orderings are fixed without regard to semantic scope. For example, *chi* 'causative' always precedes *mu* 'directional' or 'trans/cis-locative'; thus a

form like *qhawachimun* could mean either "she causes [someone] to go look", where 'cause' has scope over 'go', or "she goes to show [someone]", where 'go' has scope over 'cause'. On the other hand, a pure "semantic features" analysis predicts that orderings will NOT vary with a corresponding difference in semantic scope; but this fails because there are such contrasts. For example, *chi* 'causative' can precede or follow *na* 'reciprocal', with a corresponding difference in meaning; compare *qhawanachin* "he made them see each other" ('cause' has scope over 'reciprocal') and *qhawachinakunku* "they showed each other" ('reciprocal' has scope over 'cause').

Muysken (1986) suggests that there are three classes of verbal suffixes: *lexical* ones, which have a fixed order and are idiosyncratic in meaning (none of which I discuss here); *syntactic* ones, which are unordered and combine variably with corresponding shifts in meaning; and *inflectional* ones, which have a fixed order and no lexical meaning. This analysis is appealing because it tries to explain suffix ordering as a result of the properties of the suffixes involved. But the result is a system in which *na* 'reciprocal' belongs to the class of "syntactic" suffixes and therefore has lexical meaning, while *ku* 'reflexive', *mu* 'directional', and *pu* 'benefactive' belong to the class of "inflectional" suffixes and hence have only grammatical meaning. While this division appears to be necessary given the observed patterns of suffix ordering, it does not seem correct to describe it in terms of these particular labels and their associated properties. Moreover, under this system the ordered modal suffixes (*ku mu pu*), which often yield idiosyncratic changes in meaning, are lumped together with the inflectional suffixes encoding person/number agreement, tense, and aspect, whose effects are entirely predictable.

The system I propose is a modification of Muysken's system. The modal suffixes (M's "syntactic" suffixes plus [*ku mu pu*]) are all regarded here as derivational. They all have potentially idiosyncratic meanings and are unordered except for [*ku mu pu*], whose lexical representations indicate special ordering.

In the parser, modal suffixes are all Verb to Verb transitions. The fixed order of the [*ku mu pu*] set is derived by associating an integer (a "level") with each suffix. Each time a suffix is recognized, its level is transferred to the stem; and if the level of a candidate suffix is lower than the current level of the stem, that suffix is rejected. All the modal suffixes except for the fixed-order suffixes are level 0, permitting any order of occurrence. The fixed-order suffixes have levels 1, 2, and 3, ensuring that they follow other suffixes and occur in that order.

There remain problems with this system. While it seems to be true that the fixed-order suffixes always occur in that order relative to each other, it is apparently not the case that these suffixes always follow all the other derivational suffixes. More work needs to be done before an adequate characterization of the system is obtained. It is not difficult to find examples which contradict the descriptions given here and

in the works cited above, whether on semantic or on structural grounds. Consider the following example from (Herrero and Sánchez de Lozada (1978)), p. 258f.

(5) machu alqoncheq chillwisituta mikhorqakapurparin

mikhu	rqa	ka	pu	rpari	n
eat	honorific	reflexive	benefactive	suddenly	3sg

"Nuestro perro viejo se ha comido a nuestro pollito con avidez en un santiamén." [Our old dog ate the little chicken greedily and in a jiffy.]

Note that in this example the modal suffix *rpari* follows the fixed-order suffixes *ka (ku)* and *pu*. A description in which [*ku mu pu*] necessarily follow all other modal suffixes would not allow for this word.

8.3.2 Other derivational morphology

Compared with the complex verbal system, there is relatively little derivational morphology in nouns. There are various suffixes (*-lu, -sapa, -skiri*) which attach to either verbs or nouns and derive a noun stem with the meaning 'one who habitually does X', or 'one who has a big X'. To my knowledge, these suffixes always derive nouns which are used to refer to people. There are also nominalizing suffixes (*-na* 'thing to do X with', *-ri* 'one who does X') and verbalizing suffixes (*-ku* 'be X', *-chi* 'make X', *-ya* 'become X'). In the parser, these are handled as category transitions with various additional effects on the lexical representation of the stem in semantic and/or subcategorization features.

It is appropriate to note here that this system only accounts for the derivational processes which can be characterized by rule. It does not account for idiosyncratic meanings or argument structures, and at this point does not even check for them. A better version of the parser would look for the idiosyncratic stems and possibly disregard the compositionally regular ones in their favor.

One suffix which is classified by Herrero and Sánchez de Lozada (1978) as derivational is *-yoq*, glossed as 'owner of X', 'one who has X', or 'with X'. It appears in words like *wawa-yoq* (child-YOQ) 'pregnant', *wasi-yoq* (house-YOQ) 'landlord', and *qolqe-yoq* (money-YOQ) – either 'rich' or just 'one who has money' (any amount). The potential problem with treating *-yoq* as a derivational suffix is that it can follow the plural suffix. For example, *wasi-s-ni-yoq*[2] is also glossed 'landlord', but here it refers to someone who owns more than one house. This could be a problem for morphological analysis: the feature 'plural' is an inflectional one, since rules of syntactic

[2]The presence of the syllable *ni* in this word is discussed in section 8.4.3.

agreement must have access to that feature, and it is assumed that derivational processes do not apply to inflected words; yet the suffix *yoq* attaches to the pluralized stem.

However, the plural marker *-s* (a borrowing from Spanish) could be regarded as a derivational suffix whose effect is to add semantic number. This is plausible since it does not always trigger agreement, and sometimes imparts an idiosyncratic sense to the stem.

(6) Tata-s-ni-y munaku-wa-n.
father-pl-NI-my love-me-3sg
'My parents love me.'

The native plural form *-kuna* is available as the inflectional plural suffix: *-kuna* occurs throughout the nominal inflectional paradigm, while *-s* appears only on uninflected vowel-final stems. For example, "owners of houses" would be *wasi-s-ni-yoq-kuna*, but could not be **wasi-s-ni-yoq-s* or **wasis-s-ni-yoq-ni-s*.

8.3.3 Parsing derivational morphology in Qpop

In Qpop, lexical stems are specified with a form, a category, internal argument slots (if any), and a gloss. Derivational suffixes are specified according to the following template:

(7) [form [transition] [level] [actions] [gloss]]

Examples:
[chi [Verb Verb] [1] [addarg dobj] [gloss [cause to]]]
[ku [Verb Verb] [2] [remarg dobj] [gloss [self-]]]
[ya [Noun Verb] [1] [gloss [become]]]

The *transition* specifies the syntactic category a stem receiving the suffix must have and what category the resulting stem will have. The *level* is a condition on affixation: a level 1 suffix may not attach to a stem which has a level greater than 1. The effect of this is to get relative ordering without specifying additional states and optionality without empty transitions. *Actions* specify any effects the suffix has on the stem besides changing category and semantics, such as adding or subtracting an argument position.

As an example of how the derivational component of the parser works, consider the partial derivation of *wañuchimusaq* "I'm going to go kill [him/her/it/them]". First, the lexical stem is found:

[wañu [Verb] [gloss [die]]]

When the suffix *chi* is recognized, its representation is combined with the representation for *wañu* to give the following complex stem:

[wañuchi [Verb] [args dobj] [gloss [cause to [die]]]]

When this is combined with the representation for *mu*, the result is the following:

[wañuchimu [Verb] [args dobj] [gloss [come/go [cause to [die]]]]]

The lexical stem representation and the two complex ones are then fed to the inflectional component as candidate stems.

8.4 Inflectional Morphology

8.4.1 Finite Verbal Inflection

Quechua verbs reflect a number of inflectional categories realized as suffixes in roughly the following order:

stem—progressive—object agreement—tense—agreement—plural

Some interesting properties of the system are not apparent from this schema, however. These involve the position above labeled "agreement". The suffix appearing in this position may signal either subject or object agreement or both, depending on what occurs in the "object agreement" slot and on the person features themselves. Compare the two words shown below:

(8) qhawa wa rqa yku
watch 1 obj past 1 pl.excl
'[You/she/he/they] watched us (exclusive)'

(9) qhawa rqa yku
watch past 1 pl.excl
'We (exclusive) watched [you/her/him/them/it]'

In the first word, *wa* signals that the object is first person, and *yku* signals that the object is first person plural exclusive (excluding the addressee). There is simply

no overt subject agreement at all. In the second word, *yku* signals that the *subject* is first person plural, and object agreement is not marked. Notice that the suffix which marks first person objects (*wa*) is separated from the final agreement suffix by a tense marking suffix. Yet the interpretation of the final suffix (as subject or object agreement) depends on whether an object suffix was found. In a finite state parser, there are two options available: first, *yku* can simply be ambiguous as to whether it marks subject or object agreement. In that case, however, the information that the parser gives to the syntactic processor will be inadequate. Second, this information could be extracted by creating separate parse paths for subject-marking and object-marking occurrences of *yku*, which is undesirable.

There is no overt third person object agreement; subject agreement in intransitive verbs is exactly equivalent to subject agreement on transitive verbs when the object is third person. The paradigm for the past tense forms of the verb *apa* 'take' are given below, with agreement shown in italics. The suffix *rqa* signals past tense.

(10)

1sg	aparqa*ni*
1+2pl	aparqa*ncheq*
1pl	aparqa*yku*
2sg	aparqa*nki*
2pl	aparqa*nkicheq*
3sg	aparqa
3pl	aparqa*nku*

Now consider the verb paradigm in (11) below. Shown below are only tense and agreement suffixes, as they would occur attached to a verb stem (subject—vertical; object—horizontal); agreement suffixes are again in italics.

(11) Subject-Object Agreement (past tense)

Obj / Subj	1sg	1pl.incl	1pl.excl	2sg	2pl
1sg	*	*	*	-rqa*yki*	-rqa*ykicheq*
1+2pl	*	*	*	-rqa*ncheq*	-rqa*ncheq*
1pl	*	*	*	-rqa*yku*	-rqa*yku*
2sg	-*warqanki*	*	-*warqayku*	*	*
2pl	-*warqankicheq*	*	-*warqayku*	*	*
3sg	-*warqa*	-*warqancheq*	-*warqayku*	-*sorqa*	-*sorqacheq*
3pl	-*warqanku*	-*warqancheq*	-*warqayku*	-*sorqanku*	-*sorqacheq*

The stars represent gaps in the paradigm resulting from the fact that when the subject and object are coreferent, the verb is intransitivized with the derivational reflexive suffix *ku* and inflection is as in (10).

(11) shows that the discontinuous dependency observed above occurs all over the system. Notice that when either the subject or object argument is first person

plural it is always marked overtly, *to the exclusion of other arguments.* Note also that the second person object marker (*su*) occurs only when the subject is third person. The point I want to make here is that even a finite state system which leaves final agreement suffixes unmarked as to whether they mark subject or object must repeat suffixes along various paths simply to get the right sequences. For example, from a node (A) representing the state of a verb stem about to receive inflection, the arc reading the second person object suffix *su* reaches a node (B) which must allow an arc leaving it reading the past tense suffix *rqa*. The arc from (A) reading the first person object suffix *wa* must reach a node (B′) which must also allow an arc reading the past tense suffix. However, (B) and (B′) cannot be the same node, because the set of suffixes which may follow the past tense suffix is different in forms with *su* from the set occurring in forms with *wa*. Hence, in a finite state system the suffix *rqa* must have multiple occurrences in the network.

The obvious implication is that when the parser reaches the node following the arc reading the past tense suffix, it needs to be able to *remember* whether an object suffix was recognized in order to narrow down the set of possible continuations and to construct the right feature matrix. One way to do this is to build the morphosyntactic feature matrix as the parse proceeds and unify that set of features with the features of each candidate suffix, eliminating suffixes with clashing features.

In Qpop, inflectional suffixes are specified with a syntactic category, a morphological category transition, conditions on the current stem's features, and a set of features to incorporate into the representation. The template looks like this.

(12) [form cat transition features]

The following are sample suffix representations:

(13) [yki Verb [Vtns Vper] [[agr [sbj [[me +]
[you −]
[pl −]]]
[obj [[me −]
[you +]]]]]]

(14) [yku Verb [Vtns Vper] [[agr [[me +]
[you −]
[pl +]]]]]

Note that in the representation for *yku*, agreement features are not specified as to whether they mark subject or object features. If the object suffix *wa* is recognized, the agreement features for *yku* will unify with the features which specify the value

of *obj*. In fact, the agreement features will unify with the value of *sbj* as well, but a late filter rules out any morphosyntactic representation in which first or second person features are positively specified in both subject and object value sets.

Notice that there is a distinction between derivational and inflectional suffixes in the kind of transition made. Derivational suffixes make transitions between syntactic categories, while inflectional suffixes make transitions between inflectional states. This reflects the fact that inflectional categories do nothing but change the inflectional state of a word; they do not change the category or affect other features of the stem.

8.4.2 Infinitival Forms

Compare the first person agreement on the verb form in (15a) with that on the noun in (15b).

(15) (a)
riku na su y paq
see NA 2.obj 1sg ben
'in order for me to see you'

(b)
tura y paq
brother 1sg ben
'for my brother'

Verbs of subordinate clauses in Quechua are usually described as having been "nominalized": they are marked with case suffixes, and subject agreement is identical to possessive agreement. However, they also act like verbs: the stem resulting from NA-suffixation, unlike a lexical noun, may receive the object suffixes *wa* and *su* and must receive possessive inflection for subject agreement.[3] The choice of subordinating suffix depends on the type of clause the verb occurs in, such as a sentential adverbial, VP adverbial, or subcategorized complement, and on whether the action is realized or future/purposive. However, the agreement on subordinate verb forms is always possessive agreement. The question is how to account for the fact that a subordinated verb form is different from a lexical or derived noun in *requiring* possessive inflection.

[3]The inflectional subordinating suffix *-na* is essentially different from the derivational suffix *-na* which derives nouns from verbs. The meaning of a noun derived by the derivational *-na* is not predictable, although there are common semantic features; moreover, just as with derivational suffixes in other languages, it is not completely productive. For example, *mikhuna* ('eat-NA') is 'food', but *pichana* ('clean-NA') is 'broom', and as far as I know there is no word **asina* ('laugh-NA').

In Qpop, subordinated verb stems are fed to the nominal inflection network. A subordinated verb stem uninflected with a possessive suffix is allowed to pass through the way an uninflected noun would, but because it retains the category feature [Verb], it will be thrown out at the end because it lacks inflection.

8.4.3 Nominal Inflection and the empty morph *ni*

Inflection in nouns follows roughly the following pattern:

stem—plural—possessive—genitive—case

Nominal inflection follows the same procedure as verbal inflection. However, there is an interesting problem in the nominal inflection paradigm. Verb stems, without exception, end in a vowel. Noun stems, however, may end in a consonant, and noun suffixes (like verb suffixes) may have an initial consonant cluster. The combination of a consonant-final stem and a cluster-initial suffix (or a suffix consisting of a single consonant) results in a medial CCC or word final CC cluster. In this situation, the "empty morph" *ni* appears, with the effect of breaking up the unpronounceable cluster. Compare the forms below.

	wawa	'child'	wawas	'children'
(16)	wawan	'his/her child'	wawasnin	'his/her children'
	wawayku	'our child'	wawasniyku	'our children'

The suffixes with which *-ni* appears include the possessive suffixes (*-y, -yku, -ncheq, -yki, -ykicheq, -n, -nku*) and the case suffixes *-nta* 'through', and *-ntin* 'inclusive'. If this were all there were to it, we might be inclined to consider the appearance of *-ni* as a phenomenon motivated purely by phonotactic constraints.

However, *-ni* also occurs with the suffix *-yoq* (discussed in section 8.3.2), which is unexpected since *-yoq* has no initial consonant cluster. Thus it appears that the occurrence of *-ni* is not phonologically regular, but conditioned morphologically. In the suffix lexicon, those suffixes which occur with *-ni* are marked so that a process is triggered to insert *-ni* in case the stem is consonant-final.

Interestingly, *ni* appears even when the impossible cluster is broken up by something else.

(17)	wawaslla (*wawasnilla)	'only children'
	wawasnillanku (*wawasllanku)	'their only children'

The clitic *lla* attaches to stems; it has attached to the stem above *after* the stem has been embellished with *ni*. This is further evidence that the appearance of *ni*

is morphologically conditioned, and moreover that *ni* insertion occurs before clitics are attached. Otherwise, the placement of the clitic would do the job of breaking up the impossible consonant cluster.

An alternative way of handling *-ni* would be to include it in the lexicon of suffixes (since, after all, it is a "morph", albeit an empty one). I rejected this approach because *-ni* holds no consistent position in the inflectional paradigm; it can even occur twice in the same word.

(18) wasi–s–ni–yoq–ni–n
house–plural–NI–owner–NI–3sg
'his/her landlord'

8.5 Conclusion

Many problems remain in the current implementation of Qpop. As it is now, the inflectional feature collection procedure does not make use of a standard unification algorithm, and nothing has been said here about the way default feature values are assigned. Moreover, various aspects of the morphology have not yet been included in the implementation. One such is the filter blocking positive specifications in both subject and object agreement features for first or second person. Also, the parser does not yet recognize the suffixes described above as phrasal clitics.

The parser described here differs from previous morphological parsers in distinguishing between derivational and inflectional morphology and in making use of a memory to construct a morphosyntactic feature representation. The fragments of Quechua morphology described above demonstrate the need for something more powerful than finite state morphology; and if the output of the parser is to be useful in syntactic processing, then something more than a simple collection of morphological categories is necessary as well.

References

Alam, Yukiko Sasaki (1983). Two-level morphological analysis of Japanese. *Texas Linguistic Forum*, 22:229–252.

Albó, Xavier (1969). Review of *Cochabamba Quechua Syntax. International Journal of American Linguistics*, 37:55–61.

Anderson, Stephen R. (1975). On the interaction of phonological rules of various types. *Journal of Linguistics*, 11:39–63.

Anderson, Stephen R. (1977). On the formal description of inflection. *Proceedings of the Chicago Linguistic Society*, 13:15–44.

Anderson, Stephen R. (1982). Where's morphology? *Linguistic Inquiry*, 13:571–612.

Anderson, Stephen R. (1986a). Disjunctive ordering in inflectional morphology. *Natural Language and Linguistic Theory*, 4:1–32.

Anderson, Stephen R.(1986b). Inflection. Paper read at Milwaukee Morphology Meeting; to appear in the proceedings of the conference.

Anderson, Stephen R. (1987). Morphological theory. In Newmeyer, Frederick J., editor, *Linguistics: The Cambridge Survey (vol. I)*, pages 146–191, Cambridge University Press, Cambridge.

Aronoff, Mark (1976). *Word Formation in Generative Grammar.* MIT Press, Cambridge, MA.

Barry, W. (1984). Segment or syllable? A reaction-time investigation of phonetic processing. *Language and Speech*, 27:1–15.

Barton, Jr., G. Edward (1986). *The Computational Complexity of 2-Level Morphology.* AI Memo 856, MIT AI Laboratory.

Barton, Jr., G. Edward, Robert Berwick, and Eric Ristad (1987). *Computational Complexity and Natural Language.* MIT Press, Cambridge, Mass.

Berko, J. (1958). The child's learning of English morphology. *Word*, 14:150–177.

Bills, Garland, Bernardo Vallejo C., and Rudolph C. Troike (1969). *An Introduction to Spoken Bolivian Quechua.* University of Texas Press, Austin.

Boisen, Sean and Yoko Mimori (1986). Computational approaches to Khalkha morphology. Unpublished UCLA Manuscript.

Bradley, D., M. Garrett, and E. Zurif (1980). Syntactic deficits in Broca's aphasia. In Caplan, D., editor, *Biological Studies of Mental Processes*, pages 269–286, MIT Press, Cambridge, MA.

Burani, C., D. Salmaso, and A. Caramazza (1984). Morphological structure and lexical access. *Visible Language*, 4:348–358.

Butterworth, B. (1983). Lexical representation. In Butterworth, B., editor, *Language Production, Volume 1*, Academic Press, London.

Caramazza, A., G. Miceli, C. Silveri, and A. Laudanna (1985). Reading mechanisms and the organization of the lexicon: evidence from acquired dyslexia. *Cognitive Neuropsychology*, 2:81–114.

Chomsky, Noam (1970). Remarks on nominalizations. In Jacobs, R. A. and P. S. Rosenbaum, editors, *Readings in English Transformational Grammar*, pages 184–221, Ginn & Co., Waltham, MA.

Clements, George N. (1985). The geometry of phonological features. *Phonology Yearbook*, 2:225–252.

Clements, George N. (1986). Compensatory lengthening and consonant gemination in LuGanda. In Wetzels, L. and E. Sezer, editors, *Studies in Compensatory Lengthening*, pages 37–77, Foris, Dordrecht.

Clements, George N. and Samuel Jay Keyser (1983). *CV Phonology: A Generative Theory of the Syllable.* MIT Press, Cambridge, MA.

Cohn, Abigail (1987). Cyclic stress in Indonesian. Unpublished manuscript, UCLA.

Cole, Jennifer A.(1986). Parsky: A Parser of Finnish Inflectional Morphology. Unpublished B.A. thesis, University of California at Santa Cruz.

Kaplan, Ronald and Martin Kay (1981). Phonological rules and finite state transducers. Unpublished paper read at Winter meeting of Association for Computational Linguistics and Linguistic Society of America, New York.

Karttunen, Lauri (1983). KIMMO: a general morphological processor. *Texas Linguistic Forum*, 22:165–86.

Karttunen, Lauri and Kent Wittenberg (1983). Two-level morphological analysis of English. *Texas Linguistic Forum*, 22:217–28.

Kay, Martin (1977). Morphological and syntactic analysis. In Zampolli, A., editor, *Linguistic Structures Processing*, pages 131–234, North-Holland, Amsterdam.

Kay, Martin (1983). When meta-rules are not meta-rules. In Sparck-Jones, K. and Y. Wilks, editors, *Automatic Natural Language Parsing*, pages 94–116, Ellis Horwood, Chichester.

Kay, Martin (July 1985). Two-level morphology with tiers. unpublished paper read to CSLI workshop on finite-state morphology.

Kemply, S. T. and J. Morton (1982). The effects of priming with regularly and irregularly related words in auditory word recognition. *British Journal of Psychology*, 73:441–454.

Kenstowicz, Michael and Charles Kisseberth (1979). *Generative Phonology.* Academic Press, New York.

Keyser, S. Jay and Paul Kiparsky (1984). Syllable structure in Finnish phonology. In Aronoff, Mark and Richard Oehrle, editors, *Language Sound Structure*, pages 7–31, MIT Press, Cambridge, MA.

Kiparsky, Paul (1982). Lexical morphology and phonology. In *Linguistics in the Morning Calm*, pages 3–91, Hanshin Publishing Company, Seoul.

Koskenniemi, Kimmo (1983a). Two-level model for morphological analysis. In *Proceedings of IJCAI-83*, pages 683–85.

Koskenniemi, Kimmo (1983b). *Two-level Morphology: A General Computational Model for Word-form Recognition and Production.* Publication 11, Dept. of General Linguistics, University of Helsinki.

Koskenniemi, Kimmo (1984). A general computational model for word-form recognition and production. In *Proceedings of Coling-84*, pages 178–81.

Lapoliwa, Hans (1981). *A general approach to the phonology of Indonesian.* Volume 34 of *Pacific Linguistics series D*, Australian National University, Canberra.

Lastra, Yolanda (1968). *Cochabamba Quechua Syntax.* Mouton, The Hague, Janua Linguarum Series Practica, XL.

Laudanna, A. and C. Burani (1985). Address mechanisms to decomposed lexical entries. *Linguistics*, 23:775–792.

Levin, J. (1985). *A Metrical Theory of Syllabicity.* PhD thesis, M.I.T.

Lewis, Geoffrey (1956). *Turkish Grammar.* Oxford University Press, Cambridge.

Lopez, Barbara S. (1979). *The Sound Pattern of Brazilian Portugese (Cariocan Dialect).* PhD thesis, U.C.L.A.

Lowenstamm, Jean and Jonathan Kaye (1986). Compensatory lengthening in Tiberian Hebrew. In Wetzels, L. and E. Sezer, editors, *Studies in Compensatory Lengthening*, pages 97–132, Foris, Dordrecht.

Lukatela, G., B. Gligorijerć, A. Kostić, and M. Turvey (1980). Representation of inflected nouns in the internal lexicon. *Memory and Cognition*, 8:415–423.

Macdonald, R. Ross and Soenjono Darjowidjojo (1967). *Indonesian Reference Grammar.* Georgetown University Press, Washington, DC.

Manelis, L. and D. A. Tharp (1977). The processing of affixed words. *Memory and Cognition*, 5:690–695.

Marslen-Wilson, W. (1973). Linguistic structure and speech shadowing at very short latencies. *Nature*, 244:522–523.

Marslen-Wilson, W. D. (1980). Speech understanding as a psychological process. In Simon, J. C., editor, *Spoken Language Generation and Understanding*, Reidel, Boston.

Marslen-Wilson, W. (1984). Function and process in spoken word recognition: a tutorial review. In Bouma, H. and D. Bouwhuis, editors, *Attention and Performance X: Control of Language Processes*, Lawrence Erlbaum Associates, New Jersey.

Marslen-Wilson, W. and L. Tyler (1981). Central processes in speech understanding. *Philosophical Transactions of the Royal Society of London*, B 295:317–332.

Colé, P., C. Beauvillian, B. Pavard, L. Segui, and M. L. Zubizarreta (n.d.). L'access au lexique pour les mots morphologiquement complexes. Rapport A.T.P., CNRS (in preparation).

Cutler, Anne (1983). Lexical complexity and sentence processing. In Flores d'Arcais, G. B. and R. J. Jarvella, editors, *The Process of Language Understanding*, pages 43–79, John Wiley & Sons, New York.

Cutler, Anne (1985). Cross-language psycholinguistics. *Linguistics*, 23:659–667.

Cutler, A., J. Hawkins, and G. Gilligan (1985). The suffixing preference: a processing explanation. *Linguistics*, 23:738–758.

Cutler, Anne, Jacques Mehler, Dennis Norris, and Juan Segui (1986). The syllable's differing role in the segmentation of French and English. *Memory and Language*, 25:385–400.

Drewnowski, A. and A. F. Healy (1980). Missing *-ing* in reading: letter detection errors on word endings. *Journal of Verbal Learning and Verbal Behavior*, 19:247–262.

Dudas, Karen (1974). A case of functional phonological opacity: Javanese elative formation. *Studies in the Linguistic Sciences*, 4,2:91–111.

Emmorey, Karen (1986). Morphemic structure and lexical access. *UCLA Working Papers in Phonetics*, 63:110–124.

Emmorey, Karen (1987). *Morphological Structure and Parsing in the Lexicon.* PhD thesis, U.C.L.A.

Gajek, Oliver, Hanno T. Beck, Diane Elder, and Greg Whittemore (1983). KIMMO lisp implementation. *Texas Linguistic Forum*, 22:345–51.

Garrett, M. (1980). The limits of accommodation: arguments for independent processing levels in sentence production. In Fromkin, V., editor, *Errors in Linguistic Performance*, Academic Press, New York.

Gazdar, Gerald (1985). *Finite State Morphology: A review of Koskenniemi (1983).* Technical Report CSLI–85–32, Stanford University CSLI.

Grosjean, F. (1980). Spoken word recognition processes and the gating paradigm. *Perception and Psychophysics*, 28:267–283.

Halle, Morris (1973). Prolegomena to a theory of word formation. *Linguistic Inquiry*, 4:3–16.

Hankamer, Jorge (1986). Finite state morphology and left to right phonology. *West Coast Conference on Formal Linguistics*, 5:41–52.

Hayes, Bruce (1986). Inalterability in CV Phonology. *Language*, 62:321–351.

Henderson, L. (1985). Towards a psychology of morphemes. In Ellis, A., editor, *Progress in the Psychology of Language, Vol. 1*, Lawrence Erlbaum Associates, New Jersey.

Henderson, L., J. Wallis, and D. Knight (1984). Morphemic structure and lexical access. In Bouma, H. and D. Bouwhuis, editors, *Attention and Performance X: Control of Language Processes*, pages 211–226, Lawrence Erlbaum Associates, New Jersey.

Herrero, Joaquin S. J. and Federico Sánchez de Lozada (1978). *Gramatica Quechua: Estructura del Quechua Boliviano Contemporaneo.* Editorial Universo Ltda., Cochabamba, Bolivia.

Herrero, Joaquin S. J., Federico Sánchez de Lozada, and Luis Morató Peña (1978). *Lecciones de Quechua.* Instituto de Idiomes, Padres de Maryknoll, Cochabamba, Bolivia, Segunda Edicion.

Hillis, W. Daniel (1985). *The Connection Machine.* MIT Press, Cambridge, MA.

Hooper, Joan (1976). *An Introduction to Natural Generative Phonology.* Academic Press, New York.

Hyman, Larry (1985). *A Theory of Phonological Weight.* Foris, Dordrecht.

Jarvella, R. J. and G. Meijers (1983). Recognizing morphemes in spoken words: some evidence for a stem organized mental lexicon. In Flores d'Arcais, G. B. and R. J. Jarvella, editors, *The Process of Language Understanding*, John Wiley & Sons, New York.

Johnson, C. Douglas (1972). *Formal Aspects of Phonological Description.* Mouton & Co., The Hague.

Kaisse, Ellen and Patricia Shaw (1985). On the theory of lexical phonology. *Phonology Yearbook*, 2:1–30.

Matthews, Peter H. (1972). *Inflectional Morphology.* Cambridge University Press, Cambridge.

McCarthy, John J. (1981). A prosodic theory of non-concatenative morphology. *Linguistic Inquiry*, 12:373–418.

McCarthy, John and Alan Prince (1987). Prosodic morphology. Manuscript, University of Massachusetts and Brandeis University.

Morton, J. (1969). Interaction of information in word recognition. *Psychological Review*, 76:165–178.

Morton, J. (1979). Facilitation in word recognition: experiments causing change in the logogen model. In Kolers, P. A., M. Wrolstad, and H. Bouma, editors, *Processing of Visible Language, Vol. 1*, pages 108–156, Plenum Press, New York.

Murrell, G. and J. Morton (1974). Word recognition and morphemic structure. *Journal of Experimental Psychology*, 102:963–968.

Muysken, Pieter (1977). *Syntactic Developments in the Verb Phrase of Ecuadorian Quechua.* The Peter de Ridder Press, Amsterdam.

Muysken, Pieter (1986). Approaches to affix order. *Linguistics*, 24:629–643.

Parker, Gary (1969a). Comparative Quechua Phonology and Grammar I: Classification. *University of Hawaii Working Papers in Linguistics*, 1:65–88.

Parker, Gary (1969b). Review of *Cochabamba Quechua Syntax. Language*, 45:702–708.

Pereira, Fernando C. N. and David H. D. Warren (1980). Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artifical Intelligence*, 000:231–278.

Poppe, Nicholas (1970). *Mongolian Language Handbook.* Center for Applied Linguistics, Washington, D.C.

Poser, William (1986). Japanese evidence bearing on the compensatory lengthening controversy. In Wetzels, L. and E. Sezer, editors, *Studies in Compensatory Lengthening*, pages 167–186, Foris, Dordrecht.

Pylyshyn, Z. (1984). *Computation and Cognition.* MIT Press, Cambridge.

Rumelhart, David E. and James L. McClelland (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, Cambridge, MA, Institute for Cognitive Science, University of California, San Diego.

Schein, Barry and Donca Steriade (1986). On geminates. *Linguistic Inquiry*, 17:691–744.

Segui, J. and M. Zubizarreta (1985). Mental representation of morphologically complex words and lexical access. *Linguistics*, 23:759–774.

Seidenberg, M., G. Waters, M. Sanders, and P. Langer (1984). Pre– and postlexical loci of contextual effects on word recognition. *Memory and Cognition*, 12:315–328.

Smith, P. T. (1982). Factors affecting the perceived morphemic structure of written words. *Journal of Verbal Learning and Verbal Behavior*, 21:704–721.

Smith, P. T. and A. Groat (1979). Spelling patterns, letter cancellation, and the processing of text. In Kolers, P. A., M. Wrolstad, and H. Bouma, editors, *Processing of Visible Language, Vol. 1*, pages 309–324, Plenum Press, New York.

Stanners, R. F., J. J. Neiser, and S. Painton (1979). Memory representation for prefixed words. *Journal of Verbal Learning and Verbal Behavior*, 18:733–743.

Steriade, Donca (1987). Locality conditions and feature geometry. *Proceedings of NELS*, 17. (to appear).

Taft, M. (1979). Recognition of affixed word and the word frequency effect. *Memory and Cognition*, 7:263–272.

Taft, M. (1981). Prefix stripping revisited. *Journal of Verbal Learning and Verbal Behavior*, 20:289–297.

Taft, M. and F. Forster (1974). Lexical storage and retrieval of prefixed words. Paper presented at the Experimental Psychology Conference, Melbourne.

Taft, M. and F. Forster (1975). Lexical storage and retrieval of prefixed words. *Journal of Verbal Learning and Verbal Behavior*, 14:635–647.

Tanenhaus, M., G. Carlson, and M. Seidenberg (1985). Do listeners compute linguistic representations? In Dowty, D., L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 359–404, Cambridge University Press, New York.

Thomas-Flinders, Tracy (1981). *Inflectional Morphology: Introduction to the Extended Word and Paradigm Theory.* Occasional Papers in Linguistics 4, U.C.L.A. Dept. of Linguistics.

Thomas-Flinders, Tracy (1983). *Morphological Structures.* PhD thesis, U.C.L.A.

Thráinsson, H. (1978). On the phonology of icelandic preaspiration. *Nordic Journal of Linguistics*, 1:3–54.

Verhaar, John (1984). *Affixation in Contemporary Indonesian*, pages 1–26. Volume 18 of *NUSA: Linguistic Studies of Indonesia and Other Languages in Indonesia*, Universita Atma Jaya, Jakarta, Indonesia.

Walli-Sagey, Elizabeth (1986). On the representation of complex segments and their formation in Kinyarwanda. In Wetzels, L. and E. Sezer, editors, *Studies in Compensatory Lengthening*, pages 251–295, Foris, Dordrecht.

Wehrli, Eric (1985). Design and implementation of a lexical data base. In *Proceedings of the 2nd European meeting of ACL*, pages 146–52.

Wolff, John (1980). *Formal Indonesian.* Cornell University Southeast Asia Program, Ithaca, NY.

Yokoyama, Masako (1951). Outline of Kechua Structure I. *Language*, 27:38–67.