Morphological Processing with LR Techniques

Kang-Hyuk Lee
Information and Electronics Research Institute
Korea Advanced Institute of Science and Technology
E-mail: Khlee@csone.kaist.ac.kr

Abstract

In this paper, I present an extended two-level model using LR parsing techniques. The LR-based two-level model not only guarantees efficient morphological processing but also achieves a higher degree of descriptive adequacy than Koskenniemi's original model. The two-level model is augmented with an independent morphosyntactic module based on feature-based CF word grammar. By adopting a CF word grammar, our model is capable of dealing with complex words with discontinuous dependencies without having duplicate lexicons. It is shown how LR predictions manifested in the parsing table can help the morphological processor to minimize the dictionary lookup process.
1. Introduction

Morphology, the study of word structure, has recently been one of the most actively studied subdisciplines in theoretical linguistics. It has also gained much attention in computational linguistics during the 1980's. In this paper, I propose an extended two-level morphological analysis model using LR parsing techniques (Aho and Johnson 1974, Aho and Ullman 1977, Tomita 1986). Section 2 is a brief introduction to Koskenniemi's (1983) two-level model which has been the most prominent morphological processing model during the 1980's. Section 3 contains critical discussions of Koskenniemi's two-level formalism. In section 4, I propose a two-level morphological parser augmented with an independent morphosyntactic module based on feature-based CF word grammar. Section 4.1 discusses how feature-based grammars with CF skeletons are theoretically preferable to feature-based lexicons à la Dalrymple et al. (1987). In section 4.2, I focus on how the dictionary lookup process can be optimized by exploiting LR predictions manifested in the parsing table.
2. The Two-level Formalism

Although morphology as a subdiscipline of theoretical linguistics has attracted much attention during the past decade or so, it was virtually excluded in natural language processing until the innovative work of Koskenniemi (1983). Koskenniemi's two-level model has since been the representative computational model for morphological processing¹, as many concurrent (Karttunen and his students 1983) and follow-up works (Antworth 1990, Lee 1991) based on his model prove.

2.1 Components of the Two-level Model

As suggested by its nomenclature, the two-level model has two levels of morphophonological representations: the lexical level and surface level.² These two levels are associated by morphophonological rules which specify legitimate pairs of characters, as illustrated in figure 1.

---
¹ Of course, there have been various works on morphological parsing. For example, Hankamer(1989) reports three different approaches (including Koskeniemi's model) to morphological parsing of agglutinative languages during the 80's. However, two-level morphology has been most prominent among them.
² This subsection does not provide a detailed description of the two-level formalism. Rather, I refer the reader to Koskeniemi (1983) for full exposition of the formalism.
lexical:  s p y + s

two-level rules

surface:  s p i e s

Figure 1: The Partial Organization of the Two-level Model

The general format of two level rules is given in figure 2. CP which stands for "correspondence" refers to a lexical/surface pair. LC and RC refer to the left and right environment, respectively. OP is a logical operator which is instantiated as \( \Rightarrow \) ("if and only if") in many cases.\(^3\) What this operator says is that CP is obligatory in the given context and is possible only in that context. Notice that LC and RC also are character pairs.

\[ \text{CP} \quad \text{OP} \quad \text{LC} \quad \text{RC} \]

Figure 2: The general format of two-level rules

---

Karttunen (1983) and Sproat (1992) also are valuable sources.

3) In Koskenniemi (1983, section 2.3.9), this operator is interpreted as the combination of the two operators, namely, \( \Rightarrow \) and \( \Leftarrow \) which means "only if" and "if", respectively.
The two-level rule that legitimizes the y/i pair (or, the y/i alternation in generative-phonological parlance) in figure 1 can be put in prose as follows (cited from Karttunen and Wittenburg 1983).

\[
y/i \iff \text{CC} \quad +/\!- \quad \{i, a\}
\]

After a consonant, lexical \(y\) corresponds to \(i\) when a lexical suffix marker and any pair other than \(i/i\) or \(a/a\) follows; to \(y\) else where.

---

**Figure 3: The Y-replacement rule**

The "CC" pair stands for all the consonant pairs. "+/\!-" abbreviates the pairs consisting of the suffix marker plus any character. To sum up, two-level rules express correspondences between lexical and surface forms. This correspondence relation between two characters is a major departure from traditional generative phonology and characteristic of two-level rules.

---

4) These abbreviatory conventions make the two-level rules of a language and the corresponding finite state automata more compact and easy to read. See below.
2.2 Rules as Finite State Automata

The most innovative feature of Koskenniemi’s two-level model is the use of finite state automata (FSA)\(^5\) to encode morphophonological rules. This is why two-level morphology is often referred to as “finite state morphology.” The utilization of FSA explains why the processor is so efficient. It is well-known that finite state machines are computationally efficient and easy to implement. Finite state transducers (FSTs) behave in exactly the same way as ordinary FSAs except that they read a pair of input symbols. As shown in figure 4, a pair of characters is the input to the transducer, which is currently scanning the “y/i” pair.

![Diagram of FST processing a sequence of characters]

**Figure 4:** The acceptance of the y/i pair by the FST

---

5) More precisely, finite state “transducers” in the sense that the input string is a series of symbol pairs rather than symbols.
The English rule that expresses the y/i alternation can be depicted by a transition network diagram. Figure 5 is the graphical representation of the Y-spelling rule given in Karttunen and Wittenburg (1983).

In the actual implementation, FSTs are represented as state transition tables. Figure 6 shows the tabular form of the Y-spelling rule where a final state is indicated by a colon and a nonfinal by a period. The pair “CC” is an abbreviation for all the consonant pairs that are not “specified” in the transducer. Thus, the “CC” pair in the Y-spelling transition table stands for all the consonant pairs except for “yy”. Theoretically, “CC” includes any possible combinations of consonants such as “fv”, “zs”, “bp”, and so on. The abbreviatory conventions are not interpreted that way, however. Their interpretation varies depending on the rules of a language. For example, the schematic pair “CC” would include the pair “fv” only if it is “specified” in some other rule. The symbol “=” can be thought of as a “elsewhere condition” in phonological terms. “==” in the transition table stands for all the pairs other than the pairs specified and subsumed by more “specific” schemata (i.e., “+=” and “CC”) in a transducer. This “specificity hierarchy” is a crucial factor to interpret two-level rules/transducers. Since these notational conventions are very important to understand the two-level formalism, the reader is referred again to Koskenniemi (1983) and Karttunen (1983) for detailed description.
Figure 5: Transition network diagram for the Y-replacement rule

<table>
<thead>
<tr>
<th>state</th>
<th>1:</th>
<th>2:</th>
<th>3:</th>
<th>4:</th>
<th>5:</th>
<th>6:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 1 0 1 1 1 1</td>
<td>2 5 3 1 1 1 1</td>
<td>0 0 0 4 0 0 0</td>
<td>1 1 0 1 0 0 1</td>
<td>1 1 0 6 1 1 1</td>
<td>0 0 0 0 1 1 0</td>
</tr>
<tr>
<td></td>
<td>C y y + i a =</td>
<td>C y i = i a =</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: The state transition table for Y-replacement
2.3 Lexicon

In the two-level model, a lexicon is represented in the form of a letter tree in order to gain efficient lexical access. For example, an English lexicon that contains be, beer, believe, big and boy is roughly represented as in figure 7. The last character of each word in the lexical tree is associated with lexical entries. The b-e-e-r branch, for example, carries the entry for be at e and the entry for beer at r. The letters b and e in the third position do not have any lexical specifications because b and bee are not words in this sample lexicon. The symbols #, n-suffix, v-suffix1, v-suffix2 and a-suffix associated with the category label of each lexical item are continuation classes which refer to the relevant suffix continuation lexicons. For example, the continuation class of the category A(djective), i.e., a-suffix, would have the comparative +er and the superlative +est as its members, which allows those suffixes to be attached to an adjective. This continuation mechanism enables the morphological processor to analyze complex words such as bigger and biggest. The symbol # indicates termination, so no continuation is permitted. The continuation of a lexical formative specified in its

6) This model of lexical access originates in Knuth (1973). The letter tree is also called the discrimination network or trie. Although it is common practice to represent lexicons as lexical trees, it is often criticized due to its inappropriateness for modeling the lexical access of human processors (Forster 1976). The problem with the lexical tree model is that it predicts that it could take much less time to reject certain nonwords than to accept real words with the same length. For example, there are no English words starting with the sequence miku, so it should be possible to reject
lexical entry is how morphoactatics is described in the two-level formalism.

```
Figure 7: The lexical tree
```

nonwords such as *mikuknom* on the basis of the initial letter sequence. Since the sequence *miku* does not exist in the lexicon, the lexical search should fail at that point, requiring 4 decisions to be made. On the other hand, 8 decisions are needed to respond to the real word *American*. Thus, it is predicted that the nonexistence of *mikuknom* is detected much faster than the existence of *American*. As Forster (1976: p.260)’s experiment shows, however, nonexistent words typically take longer time to reject than existing words take to accept.
3. Problems with Morphotactics in the Two-level Model

This section discusses problems with the two-level model from both computational and theoretical perspectives and then describes an enhanced version of the two-level model which incorporates an independent morphosyntactic component. It is shown how the LR parsing table contributes to optimizing the dictionary lookup process.

3.1 Computational Problem

Acclaimed for its efficient morphological processing, the two-level model has been criticized for both theoretical and computational reasons. Gazdar (1985) points out inadequacies of the two-level model on mathematical grounds. Based on the classical 3SAT problems in computational complexity theory, Barton et al. (1987) proves that the two-level model is NP-hard. 7 Sproat and Brunson (1987) argue that there are many morphological phenomena that cannot be covered by the two-level model without altering it significantly. Byrd et al. (1986) cast doubt on the capability

7) Koskenniemi and Church (1988) is the response to this argument.
of the model in handling derivational affixation.

Although the two-level model has been plagued by both theoretical and computational problems, little has been discussed with respect to its morphotactic problem that causes serious runtime overheads. As briefly mentioned in the previous section, in the two-level model, morphosyntactic (or morphotactic) descriptions are expressed in each lexical entry in terms of the so-called continuation class. For example, a verb root would have, say, *verb-suffix* as its continuation class which comprises morphemic categories such as PRESENT (+s), PAST (+ed), PAST PARTICIPLE (+ed), PROGRESSIVE (+ing), etc. Each of these morphemes constitutes a continuation lexicon. Thus, a continuation class can be thought of as a set of lexicons. As noted by Barton et al. (1986), the reason for subdividing the dictionary this way is that in the two-level model, the continuation class mechanism is the only means to describe morphotactics, i.e., co-occurrence constraints among roots and affixes. For example, it would not be possible to capture contrasts such as *sleeper/*sleeped and *believer/believed* unless the verb suffixes -er and -ed are listed in separate lexicons. In other words, if they are in the same lexicon, it would be impossible for the continuation class of the verb *sleep* to include the agentive suffix -er but not the past suffix -ed (cf. Karttunen and Wittenburg 1983).

The computational complexity of the continuation class mechanism is bounded to $O(m \times n)$ where $n$ is the number of morphemes composing a morphologically complex word and $m$ is the number of sublexicons the continuation class of each morpheme can points to,
as shown in figure 8. The reason for this is simply that there are \( n \)-possibilities of searching sublexicons for each morpheme in the input string.\(^8\) This assumes that there are no ambiguous morphemes in the dictionary. If there are ambiguous morphemes exemplified by English suffixes such as +ed and +ing, some additional cost should be paid for backtracking.\(^9\) While trying to analyze the complex word *surprisingly*, for example, it is plausible for the parser to recognize +ing as the gerundial suffix which cannot be followed by the adverb-making suffix +ly. In that case, the parser has to backtrack to reanalyze it as the adjective-making suffix. At any rate, the continuation class mechanism itself does not reveal any serious computational complexity.

\[
\begin{align*}
M_1 & : L_1^1, L_2^1, L_3^1, \ldots, L_m^1 \\
M_2 & : L_1^2, L_2^2, L_3^2, \ldots, L_m^2 \\
M_3 & : L_1^3, L_2^3, L_3^3, \ldots, L_m^3 \\
& \vdots \\
M_n & : L_1^n, L_2^n, L_3^n, \ldots, L_m^n
\end{align*}
\]

Figure 8: Hypothetical continuation classes

---

8) This complexity result does not consider false segmentations which lead the processor to search a lot of lexicons in vain. For example, in analyzing the word *spel*, the morphological analyzer could match *sp* in *spel* with *spy* in the root lexicon, since English has the y-spelling rule which allows the "y/i" pair. Since *spy* is both noun and verb, the analyzer will exhaust all the possible sublexicons (i.e., noun suffixes and verb suffixes) before it gets to the "li" pair. This obviously slows down the system quite a bit. However, it is not a morphotactic problem. Barton et al. (1987) discuss a dictionary organization that reduces search time and space caused by false
Nevertheless, the continuation class mechanism in the two-level model could cause serious runtime overheads. This is mainly because each of the lexicons in a continuation class is traversed separately. For example, after matching *surprise*, in the complex word *surprisingly* with the corresponding lexical string in the dictionary, the morphological processor has to search each continuation lexicon until it finds the suffix *-ing* which properly matches the surface string currently being processed. It is quite possible that matching *-ing* against the continuation lexicons could succeed only after exhausting all incompatible lexicons. Given that the basic operation in the two-level model is to match one level of characters with the other level of characters under the guidance of two-level rules, futile searches of lexicons would be a serious bottleneck for morphological analyses.

### 3.2 Theoretical Problem

As pointed out in the literature (Karttunen 1983, Byrd *et al.* 1986), there are serious defects in the two-level model with respect to morphotactics, which are theoretical rather than computational (both in some cases). As discussed above, the morphosyntactic segmentations.

9) Following Karttunen (1983), I assume that the processor operates in the top-down depth-first manner.

10) The orthographic change, i.e., the deletion of *e* in *surprise* is to be handled by the morphographemic rule encoded as a finite state transducer.
constraints of the two-level model are described by specifying for each lexical item the class of morphemic items that can be attached to it (in the form of a finite state machine). This simple-minded approach collapses when there is the need for expressing discontinuous dependencies, i.e., cooccurrence constraints between morphemes that are not contiguous to each other (cf. Kauttunen 1983:178-181, Antworth 1991:12-13). For example, let us consider how English morphotactics can be described to allow derived words such as *entailment* and *ennoblement* but prohibit nonwords such as *tailment* and *noblement*. Obviously, the derivational suffix *+ment* should combine with verbs such as *entail* and *enoble*, but not adjectives such as *noble* or nouns such as *tail*. In the two-level model, however, this restriction cannot be expressed without adjectives and nouns duplicated in the lexicon, costing a great deal of memory resources. If they are defined to have a single entry, nonwords such as *tailment* and *noblement* would also be accepted as grammatical.

Another problem is that since morphotactics in the two-level model is encoded in terms of finite state transitions, it is not possible to properly group the morphemes involved in a morphologically complex word, i.e., to assign its morphological structure. For example, the word *disorganize* might be decomposed to have the internal structure in (1),

(1) $[u \text{ dis } [u [n \text{ organ } ] \text{ ize } ] ]$

In most implementations of the two-level model (e.g. Koskenniemi
1983; Karttunen et al. 1983; Antworth 1991), the structure of a word is flattened out, just listing the category labels of the morphemes comprising the word. Thus, the gloss of the word disorganize would be something like Vpref+Noun+Vsuf (i.e., verb prefix, noun root, and verb suffix). Since this flat sequence of morpheme labels does not express the structure of the word, we are also unable to determine the class (i.e., part of speech) of the entire word. Unless we are solely concerned with segmentations of words and phonological changes, representations of this kind are hardly useful for natural language processing systems which take the output of the morphological analyzer as the input of the syntactic parser. In order for the morphological processor to be more practical, some modifications are necessary to make it serve as the front-end to the syntactic parser.

4. Extending the Two-level Model

As discussed in the above two subsections, the morphotactics of the two-level model suffers from both computational and theoretical problems. This section attempts to remedy morphosyntactic problems with the two-level model. In order to achieve this, I separate the morphosyntactic component from the lexicon and adopt an independent morphosyntactic scheme which is linguistically enriched. It is also shown how LR predictions
manifested in the parsing table can help the morphological processor to minimize the dictionary lookup process.

4.1 Dispensing with Continuation Classes

To solve morphotactic problems with the two-level model, I propose to remove the continuation class mechanism from the lexicon and set a separate level for morphotactics that allows for more enriched descriptions about morpheme combinations and morphosyntactic properties of combinations. This separate morphosyntactic component is based on context-free word grammar à la Selkirk (1982) augmented with feature structures. The phonological (or morphographemic) component (i.e., a collection of two-level rules) is only responsible for segmenting the input string and returning the feature structure of each morpheme involved in the input string without indicating how they are structurally grouped. With this separate morphosyntactic component, it is easy to describe noncontiguity of morphemes. For example, the lexicon and two-level rules will segment the word disorganize into dis+organ+ize with each of these morphemes associated with the relevant feature structure. Then, the morphosyntactic parser will give the full-fledged feature structure, i.e., the feature-based lexical entry, via unification and word-grammar shown in figure 9.

\[ V \rightarrow N \quad V_{suf} \]
\[ V \rightarrow V_{pref} \quad V \]

Figure 9: Word grammar for disorganize
It should be noted that it is not a new idea to use a feature-based lexicon in order to build up word structures by means of unification. Dalrymple et al. (1987) discuss an implementation of the two-level model called "DKimmo" in which the lexical entry for each lexical item is a feature-value matrix. And the feature structure of a morphologically complex word is built by unifying all the feature structures of morphemes composing the word. This approach also obviates the need for specifying continuation classes because the compatibility between morphemes is to be checked through unification. However, Dalrymple et al. do not make use of word grammar of any sort, thus discriminated from the approach based on the context-free word grammar. This means that feature-based lexical entries must be defined not to have any feature conflicts. This does not seem to be problematic for category-preserving affixations such as love+s. Unifying the feature structure of the verb love with that of the suffix +s will produce the feature structure with all the agreement information fully specified, as shown in figure 10 and 11.

\[
\text{love: } \left[ \begin{array}{c}
cat: v \\
nsubjcat: \{np, np\}
\end{array} \right] \\
\text{+s: } \left[ \begin{array}{c}
agreement: \{number: singular\}
\end{array} \right]
\]

Figure 10: Lexical entries for love and +s
loves: $\left[ \begin{array}{c}
cat: v \\
subcat: \{np, np\} \\
agreement: \begin{array}{c}
number: singular \\
person: third
\end{array}
\end{array} \right]$

Figure 11: Feature-value matrix for loves

Simply unifying morphemes composing a morphologically complex word, however, is not sufficient enough to deal with important morphological behaviors. As discussed in the morphological literature (Lieber 1980, Williams 1981, Selkirk 1982 among others), morphological structures are endocentric, i.e., headed by a particular morpheme. The headedness of morphological structures is crucially related to “feature percolation”. Williams (1981) argues that the features of a morphologically complex word are inherited from the head where the head is the rightmost morphological constituent of the word. This formulation, called “Righthand Head Rule”, was primarily devised to capture the generalization that what determines the properties (most importantly, part of speech) of a morphologically complex word is suffixes (and the rightmost constituent in the case of compounds).\textsuperscript{11} For example, we might

---

11) It is easy to falsify this generalization. In the word *enoble*, for example, it is the prefix *en* that turns it into a verb, thus making the left-handed prefix the head. It is not the purpose of this section to discuss the validity of Williams’ formulation, but to point out the importance of the notion of head. See Selkirk (1982) for some modifications of the original formulation and DiSciullo and Williams (1987) for the more recent version.
have the structure in figure 12 for the derived word *shooter*. I assume, following Gazdar *et al.* (1985), that syntactic features such as *N* and diacritic features such as *person* belong to the class of head features. Being head features, all the feature-value pairs of the suffix *+er* except for the *type* feature are percolated up to the mother category.

If we adopt the scheme described in Dalrymple *et al.* (1987) in which the structure of a complex word is built by simply unifying the feature structures of the morphemes, the feature structure in figure 5.12 would not be obtained due to the feature crash between the verb root *shoot* and the suffix *+er*.

It is possible to describe category-changing affixation if the lexicon contains null categories which are phonologically empty but carry syntactic and diacritic information. A similar method is exploited by Karttunen and Wittenburg (1983) in their description of English, though they do not use feature-based categories. For example, the verb *shoot* can be defined to have no feature-value pairs or to have feature-value pairs without information determining the category status. If it is suffixed, its category will be determined by the suffix through unification.

If it is not suffixed, it will be instantiated as a verb by inheriting the feature structure of the compatible null category. Another way of saying it is that if the verb root *shoot* is suffixed by the compatible null suffix, the unspecified (or underspecified) verb root will unify with the feature structure of that null suffix containing information necessary for defining verbs. However, not only does this method
miss the linguistic generalization of affixation but also seem to be ill-suited for explaining noncontiguous morphological structures. To analyze the complex word \textit{ennoblement}, for example, the parser has to be silent about the intermediate analysis \textit{en+noble} because it is followed by the noun suffix \textit{+ment} which will finally determines the category of the word and other information pertaining to nouns. This scheme, again, leads to the proliferation of lexical entries let alone its counterintuitiveness.

4.2 Optimizing lexical search with LR-predictions

Incorporating feature-based word grammars into the two-level
model empowers us to describe morphotactics in a more enriched way. It also enables us to avoid runtime overheads caused by search through each of the sublexicons in a continuation class because with an independent module of morphotactics, we do not need to have multiple lexicons. All morphemes are entered into a single lexicon. There is a tradeoff between using multiple lexicons and using a single lexicon, however. Although a separate morphotactic component allows us to have a merged lexicon (with more adequate descriptions of morphological phenomena), it could fruitlessly expand lexical searches that would have been cut down in an earlier stage when lexicons were subdivided (Baton et al. 1987).

Since the LR method is known to be highly efficient, it is natural to adopt it to process CF-style word grammars. However, the problem with prolonged dictionary searches that arises from having a single lexicon remains. To solve this problem, I propose to make use of the parsing table as a pseudo-continuation device. To illustrate how this method works, let us consider the sample word grammar and lexicon shown in figure 13. The parsing table is given in figure 14. The dictionary is organized in such a way that each class of morphemes is rooted in a preterminal symbol of the word grammar. Though the dictionary in figure 13 is presented as CF rules for the sake of exposition, it should be taken to be a set of lexical trees each of which is indexed by (or hashes on) a preterminal symbol. A set of lexical items surrounded by curly brackets can be thought of as a lexical tree. Note that subdividing a dictionary this way is not identical with the original dictionary format based on the continuation class
mechanism in that the same class of morphemes is structured in a separate lexical tree (e.g., verb suffixes are embedded in a single lexical tree rather than listed as separate sublexicons.)

(1) \( N \rightarrow N_{stem} \quad N_{suf} \)
(2) \( V \rightarrow V_{stem} \quad V_{suf} \)

\[ N_{stem} \rightarrow \{book, boy, \ldots\} \]
\[ V_{stem} \rightarrow \{die, kill, \ldots\} \]
\[ N_{suf} \rightarrow \{+s\} \]
\[ V_{suf} \rightarrow \{+s, +ed, +ing, \ldots\} \]

Figure 13: A sample word grammar and dictionary

<table>
<thead>
<tr>
<th>State</th>
<th>( N_{stem} )</th>
<th>( V_{stem} )</th>
<th>( N_{suf} )</th>
<th>( V_{suf} )</th>
<th>$$$</th>
<th>( N )</th>
<th>( V )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>sh2</td>
<td>sh4</td>
<td></td>
<td></td>
<td>$$$</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>acc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>sh5</td>
<td></td>
<td>acc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>acc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>sh6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>re1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>re2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 14: The parsing table for the word grammar in figure 13

With the dictionary organized this way, all that is needed to minimize the lexical lookup process is a mechanism to identify the possible morphemic category that can follow the previously processed morpheme so that the parser immediately picks out a lexical tree indexed by that category symbol. Roughly, this amounts
to saying that we need a lookahead symbol to find out what comes next at a particular point. For example, after processing \textit{kill} in \textit{killed}, which means that we have segmented the verb root \textit{kill} out of the input string and have pushed \textit{Vstem} onto the stack, migrating to the state vertex 4, the parser has to scan the rest of the input string to find the following morpheme. If the lexicon had not been subdivided as in figure 13, the parser would have blindly searched through the whole lexicon until it has matched the rest of the string with the morpheme \textit{+ed}. This expensive lexical lookup process can be remedied by utilizing the parsing table which is constructed by looking ahead at grammar symbols. It requires a slightly different parsing process from sentence-level parsing in which the table lookup is done by the current state vertex and the grammar symbol obtained by scanning the next input word. Rather than traversing the lexicon based on the characters following \textit{kill}, the morphological processor will look up in the table at the vertex 4 first. According to the parsing table in figure 14, there is a shift action (\textit{i.e., sh6}) with the morphemic category \textit{V}su\textit{f}. This tells that the next segmented morpheme from the remaining input string should be a verb suffix, thus enabling the parser to pick out the exact lexicon indexed by \textit{V}su\textit{f}. Otherwise, the parser would have to search through each lexicon. By making use of predictions in the parsing table as pseudo-continuation constraints, we can get around the lexical lookup problem arising from eliminating the continuation class mechanism from the two-level model.

In sum, the LR method guarantees efficient parsing with CF word grammars that are incorporated into the two-level model to give
more enriched descriptions of morphotactics. It also turns out to be well-suited to optimizing the dictionary lookup process, thus saving the loss of efficiency that could arise from using a separate morphotactic component.

5. Conclusion

In this paper, I have discussed morphological parsing which has been an actively studied area in natural language processing during the 1980's. Outstanding problems with Koskenniemi's (1983) two-level model for morphological parsing have been discussed, focusing on morphotactics. An alternative approach to morphosyntactic descriptions for the model has been proposed which replaces the continuation-class mechanism with independent CF word grammars. It has been shown how LR predictions manifested in the parsing table can help the morphological processor to minimize the dictionary lookup process.
References


Shieber, S. 1986. An Introduction to Unification-Based Approaches to Grammar. CSLI, Stanford, CA.

Cambridge, MA.
Morphological Processing with LR Techniques

본 논문은 LR파싱기법을 이용한 확장된 두단계(two-level) 형태소분석 모델을 제시한다. LA기법을 이용한 두단계 모델은 효율적 형태소분석 뿐만 아니라 Koskenniemi(1983)의 모델보다 형태론적 현상에 대한 보다 높은 기술성(descriptive adequacy)을 획득한다. 이를 위해 두단계 모델은 자질기반의 문맥자유문법(feature-based CF grammar)에 근거한 독립적인 형태/동사 모듈에 의해 확장된다. 문맥자유문법에 근거한 단어문법(word grammar)을 채택함으로써 확장 모델은 하위사전의 중복현상을 피해하면서 비연속적 의존관계(discontinuous dependencies)를 가지는 복합어 등 을 처리할 수 있다. 또한 파생테이블에 명시된 LR 예측은 형태소분석기로 하여금 사전담색시간을 줄일 수 있도록 도와준다.