A Two-level Morphological Analysis of Korean

Tae-Yeoub Jang
tyjang@cstr.ed.ac.uk

Abstract

A method of automatic morphological analysis of Korean words using two-level rules of phonology and morphology is described. Three main parts, lexicon, rules, morphotactic constraints, are built with the two-level morphological tool 'PC-Kimmo' and are loaded to work interactively. Rules, core of the analysis, are first written by hand and then compiled to have the format to work on a finite state machine. All the corpus internal words are correctly analysed.

1 Introduction

In this paper, I describe a way of automatically analysing Korean word structures using two-level description of phonological and morphological rules (Koskenniemi 1983). The aim of the study is to provide Korean speech recognition and synthesis systems as well as syntactic parsers with more effective information on word internal structure of Korean words.

Three main parts play crucial roles. Firstly, spelling rules are described in the format of two-level morphology. The rules include processes of lexical phonology which are accompanied when frequent affix agglutination and compounding are taking place. Secondly, morpheme sequences will be restricted properly. The primary morphotactics is dealt with by lexicon component, but more detailed constraints are implemented by the features attached to the individual lexical items. The other part is the lexicon itself which is designed to contain not only all the relevant word roots and affixes but the sub-divided category names, part of speech, and the features necessary for the morphotactic constraints just mentioned.

Development of each component in terms of PC-Kimmo (Antworth 1990; Version 2.1) package, the implementation tool for two-level morphology, will be described in detail preceded by some background information on the usefulness of the analysis, the Korean language, and its Romanisation.

2 Backgrounds

2.1 Uses of morphological analysis

Lots of Korean content words, whether spoken or written, are composed of more than one single morpheme. Nouns, for instance, usually have case markers and verbs are compelled to have at least one bound morpheme agglutinated to a root so as to be grammatical.

Most leading Korean recognition systems are not using a word list and its corresponding dictionary which contain properly decomposed word internal items. As a consequence, a large number of words the root morphemes of which are identical one another have to be listed as they were disregarding the homogeneity of the words. For example, a single verb stem of Korean can generate more than 40 derived words by agglutinating separate affixes, mostly suffixes. When some of these words, if not all, are to be used within a corpus of a recognition system each of them used to be listed in the vocabulary list. The effect is that the announced vocabulary size of the system may look quite large but in reality the system can virtually cover relatively small amount of items, exaggerating performance and coverage of the system.

Text-to-speech synthesis system also needs to make use of the morphological information in establishing the accurate letter-to-sound rules of Korean. Korean is generally known as one of the languages whose phonetically driven scripting system makes it possible to directly convert words into sound symbols fairly easily. However, some words cannot completely be mapped into correct symbols without having their morphemes individually identified. Some affixed morphemes and Sino-Korean words, in particular, show different phonological behaviour against some phonological rules and they should be specified in the pre-processing lexicon unless proper morpheme analysis reveals their internal structures.

2.2 Word Structure of Korean

Word is the basic unit where the morphological analysis is performed. Korean words are relatively easily detected from the text since a space is supposed to be placed between them. [1]

Korean is a typical agglutinative language whose words are composed of at least two or more morphemes whose boundary inside the word is in most cases clear. Consider (1) for example.

(1) *saengkak+ha+si+eoss+eoss+kess+ta*

saengkak	Noun-stem meaning ”Thought”
ha	Verb-Stem meaning ”Do”
si	Honorific
eoss	Past 1
eoss	Past 2
kess	Conjecture
ta	Declarative Ending Suffix

Each morpheme has its own meaning and form, which makes the distinction of the boundaries relatively clear. Having clearer morpheme boundaries than the words of inflectional languages does not necessarily mean that there are only a small number of phonological processes when affixes are agglutinated. Rather, Korean words have plenty of morpho-phonemic rules and their exceptions which make morphemic analysis difficult.

2.3 Romanisation

The final preliminary step for the morphological analysis is putting Korean type-script characters called *Hangul* into Romanised ones. This procedure is necessary not only for making

[1] the exceptions are noun compounds. A noun compound consisting of less than four nouns is normally written without any spaces between them but a compound with more than four free morphemes can optionally be expressed with one or more spaces between individual nouns within the word for better readability. It is impossible to identify that kind of word word as a single word item only from the existence of spaces. A higher level syntactic analysis may have to be used as well. The case is beyond the scope of current research.

relevant files fit for the analysis tool PC-Kimmo but for increasing readability of the implemented rules and lexicon.

A *C++* program is created to automatically convert the 8 bit 2-byte Hangul codes named KSC-5601 into English alphabet characters. [2]

The core of Romanisation is a technique to convert 16-bit(8-bit 2-byte) characters into 7-bit single byte ones. As each precomposed Hangul character represents a syllable it also contains the separate information on sub-syllable elements like onset, nucleus, and coda. By using bitwise operator each 5 bits are designated to have each information of three syllable elements, and the other one bit (MSB) is reserved to check whether Hangul will be used or not. [3] And then each 5-bit code range can refer to onset, nucleus, and coda of a syllable and final mapping to corresponding Roman character is performed based on ready-made character conversion table.

The criteria of mapping from Hangul into Roman character is not simple and straightforward because there is no direct one-to-one correlation between the segments of each writing system. There have been heated debates on which version of Romanisation mapping is the best way to transliterate Hangul characters into Roman ones amongst linguists and Hangul researchers home or abroad. In this program, I followed Hangul Romanisation Standard agreed between both South and North Korean authorities in 1992. Only one exception is for a Hangul alphabet character called *li-eul*, which I expressed as a single form ”l” rather than distinguishing two, by ”r” for onset and ”l” for coda as in the original Romanisation agreement. I didn’t separate them in the sense that Romanisation itself is to be done in the level of phonology not of surface phonetics, and that I assume the ”l” sound is the underlying form of the the Korean alphabet ”li-eul” though I will not delve into this matter further in this paper.

Some Hangul consonants and vowels are not expressible via a single Roman character. For example, there are at least eight Korean monophthong vowel characters, [4] while only five (a , e, i, o, u) are used in Roman alphabet. It means a few vowels need to be expressed by complex form with two Roman characters. In such case, a problem can arise with regard to syllabification.

[2]In spite of frequently reported inconveniences the KSC-5601 is to date known as the national standard character code for information exchange since it is declared as such by Korean government.

[3]This is necessary because Hangul document itself before Romanisation can also contain Roman alphabets like acronyms, academic terms, or proper names.

[4]Some speakers have nine monophthong system pronouncing the diphthong ”we” as a monophthong ”oe”

For instance, a vowel sequence with only two Hangul vowels can be described as VVV form, which might be ambiguously interpreted as VV-V, or VV-V. To prevent this confusion, syllable boundary will always be inserted between two syllables in the form of hyphen, which in turn will easily be removed whenever necessary whether in synthesising or analysing words.

3 Two-Level Rules

3.1 Generative vs. Two-Level Rules

Though two level rules are in a sense similar to the rules of standard generative phonological theory in that they both use the notion of lexical forms, surface forms and relevant environments for the formalisation of rules, crucial differences lie in two aspects. First, two level-rules don't have any concept of derivation. They just state certain correspondences between a lexical form and its surface form. Consequently, rules are applied in parallel rather than in sequence and no specification of rule ordering is necessary unlike the rules in generative phonology. Second, two level rules are bidirectional while generative rules are unidirectional. It implies that two level rules can be used not only to analyse the morphological structure of words but to synthesise individual morphemes into a word, which makes the computer application easy and effective.

There are also disadvantages of the two-level rules. Above all, it is quite hard to both read and write correct forms of the rules. This is actually a trade-off for the nature of unnecessary rule ordering. It implies that every single rule should not conflict with every other rules for the successful application of that rule. One of the major criteria of generative phonology in evaluating acceptability of phonological rules, "independent motivation" seems to completely irrelevant in the two-level rule system. Rather, it goes the other way. All the rules should be written considering the environments of other rules to avoid the explicit or implicit rule conflicts.

Another problem of current two-level rules is difficulty of capturing the naturalness, which is also an important way of justification the existence of rules in generative phonology. This limitation is closely related with the unavailability of the distinctive features. Character class definition called 'Subset' of two-level rule description is an assistant method to recover that kind of deficiency, but it is still too restricted in use to capture the homogeneity of the phonemes.

3.2 Rule Format and Compilation

(2) is an example of a simple shape of two-level rules.

(2) A:B <=> ___ C:D

The rule states that lexical A is realised as surface B always and only in front of the lexical C which is realised as D on surface. The rule is made up of three parts. The first part is the "correspondence", "A:B" of the rule (2). It says how lexical form is realised at the surface level. The second part is "operator", which determines the nature of the rules and relation between the operator and environments. There are four types of operators: =>, <=, <=>, and /<=. Roughly each one means "only but not always", "always but not only", "always and only", and "never". The last operator is useful for describing a specific constraints or exceptional cases. Finally, the environment part looks similar to that of generative phonology except that the conditioning segments are also expressed as correspondences between two levels. In case there is no difference between two forms it can be written with one symbol. That is, "C" is an alternative expression of "C:C".

Thus created rules should be compiled into "state transition tables" in order that they can work in the finite state automaton (FSA, henceforth) which is designed to process well-formed strings of a regular language. The work is initially done by a two-level rule compiling tool called "Kgen (Beta version 0.2, programmed by Nathan Miles 1991)". But as the program has some known defects in fully supporting the later version of analysis tool PC-Kimmo, hand correction of the state tables was performed whenever necessary. (3) is an example of a rule and its compiled state table which was created for the Korean morphological analysis.

(3) "eu" Vowel Deletion Rule
eu:0 => _ −:@ +:0 @:eo

	eu	–	+	@	@
	0	al	0	eo	@
1:	2	1	1	1	1
2.	0	3	0	0	0
3.	0	0	4	0	0
4.	0	0	0	1	0

The meaning and role of this rule will be described in the following section, but we can see that the operator ”=>” indicates this rule applies optionally. There are four states and five correspondences including default correspondence (@:@) are relevant. State 1 is the only final state being marked with a colon and the others are non-final states and marked with a dot symbol.

4 Rules Component

This section shows how Korean morpho-phonemic rules are described. The rules are created to successfully analyse all the corpus internal complex words made up of two or more morphemes. It means that although the rules illustrated here are not sufficient for accounting for all the processes of Korean lexical phonology, other necessary rules can easily be added when new words are supplemented to the lexicon file.

Most of the rules I wrote below are about noun and verb affixation since lexical phonology and morphology of the two part-of-speech categories and their interaction are what make the Korean word formation complicated, and what produces lots of exceptions and problems.

Before rules are described, at the beginning of the rule file, all the Romanised alphabetical characters and other symbols and diacritics such as boundaries used in rule description should be declared. In my Romanisation program, 19 consonants, 2 glides, 8 vowels are used and they are specified as in (4) along with other symbols.

(4) ALPHABET
k kk n t tt l m p pp s ss c cc ch kh th ph h ng

w y a eo o u eu i e ae

NULL 0
ANY @
BOUNDARY – + #

Null symbol is necessary for the rules of insertion or deletion of segments or boundaries, and symbol "@" is used for specifying default correspondences and elsewhere effects. The last three symbols are boundaries for syllable(-), morpheme(+), and word(#). [5]

One more procedure is needed for the efficient description of rules. As mentioned in section 3.1, PC-Kimmo is not smart enough to allow natural classification of segments by way of distinctive feature notations. The supplementary device is "subset" notation, which is designed to contain multiple sounds in a single expression. The subsets used for the current rule description are as follows:

(5) Subset Declaration

SUBSET	C	k kk n t tt l m p pp s ss c cc ch kh th ph h ng
SUBSET	G	y w
SUBSET	V	a eo o u eu i e ae
SUBSET	Pv	a o
SUBSET	IOU	i o u
SUBSET	EAe	e ae

The next step is to write actual rules and transition tables. The first rule is supposed to be default correspondences which specify the way lexical forms are realised as surface forms whenever

[5] As is already mentioned earlier, without syllable boundaries some ambiguous expressions will take place due to lack of one-to-one mapping between Korean segments and Roman alphabets caused by the different number of segments of two languages. For instance, an imaginary word like *kaekkak* can be syllabified as any of 6 ways: *kae-kkak*, *kaek-kak*, *kaekk-ak*, *ka-ek-kak*, *ka-e-kkak*, *ka-ekk-ak*. Another general reason to put a syllable boundary between two morphemes is that as each Hangul character represents a single syllable on its own, Romanisation without any syllable boundary may cause a loss of information which is useful for various purposes.

they are not a part of rule description know as feasible pairs. A Default correspondence of morpheme boundary is ”+:0” so that it never shows up at the surface level and all the others do not show any difference between the forms at two levels.

Currently nineteen rules are written for analysis function and a few of them are explained in detail for illustration.

In Korean, there exists surface alternation of suffix initial vowel *eo*. The surface form *a* appears when the vowel of the preceding syllable is *a* or *o* which are frequently called as ”positive vowels”, while *eo* appears in the other environments. [6]

(6) *a/eo* alternation

Lexical Form	Surface Form	Glossary
meok+eoss-ta	meok+eoss-ta	”eat”+Past+DecFin
meok+eo-la	meok+eo-la	”eat”+ImpFin
po+eo	po+a	”see”+ImpFin
cap+eoss-ta	cap+ass-ta	”hold”+Past+DecFin

The rule for this process is written as (7).

(7) RULE 3
eo:a <=> Pv:@ (C:@) -:@ +:0 (0:w) _

This rule states vowel *eo* is realised as *a* only and always in the given environment. The subset ”Pv” means either of two vowels *a, o* as prespecified in (5). Operator ”<=>” indicates the condition ”only and if only”. ”-:@” correspondence in the environment is used because all the syllable boundaries of Korean written words are identifiable and because some rules are sensitive to the syllable boundary information.

[6]Some efforts have been made by generative phonologists to find a plausible distinctive feature which includes only two vowels *a* and *o* as a natural class among others, though it is still controversial. See Chung (1994:299) for more discussion and other literature sources.

One of the reasons I regard the lexical form of past tense pre-final suffix +*eo* rather than +*a* is as follows: As we have already seen in the example (1) in section 2.2, double past tense of a verb is possible in Korean by appending past suffix once more to the original past tense suffix. In that case, the surface form attached later is not +*ass* but +*eoss*. Thus we can regard this form as lexically specified form of Past tense and *ass* as only a surface form.

We need another rule to explain the surface forms *seo*(”stand”) whose lexical form should be *seo*+*eo*. That is, a rule to delete one of two identical vowels is necessary. Here, I make the stem vowel *eo* to be deleted rather than the second suffix vowel on the ground that there are other independently motivated rules (rule 5 and 7 in the list (11)) which deletes the the stem vowels anyway.

(8) RULE 9 (version 1)
eo:0 <=> _ -:@ +:0 eo

A revision should be made to include another *eo* deletion that forces the lexical *pe*(”cut”)-”*o* and *nae*(”put out”)-*eo* to realize as *pe* and *nae* optionally. The process does not seem to be the same category with the rule (8), but it should be expressed together in a single rule because of the fact that the PC-Kimmo forces the same correspondences to be expressed in the same rule formalisation, which is one of a few functions of the tool to be improved in the future. At any rate, the rule is modified as (9).

(9) RULE 9 (version 2)
eo:0 => _ -:@ +:0 eo | [e|ae] -:@ +:0 _

The bracket symbols means any of the items within them divided by vertical bar (|) can be placed on that position. Note the optionality of the rule by using the operator ”=>” to correctly allow both *pe* and *pe-eo* forms. But the problem arises as the illegal surface forms *seo-eo* which was correctly suppressed by the previous version (8) can now be produced owing to the changed operator. This can be resolved by adding another rule which simply prevents the occurrence of such illegal forms.

(10) RULE 10
eo:eo /<= _ -:@ +:0 eo

As mentioned in section 3.2, the operator "/<=" plays a role of a constraint suppressing specific correspondences in the given environment.

(11) is the list of rules created in this way including the final version of the *eo* vowel deletion rule with a little further modification. Again, all the rules are interacting one another and there is no rule ordering among the rules, which may result in the fact that some rules are hard to identify without a careful investigation of other rules in the list.

(11) List of Rules

RULE 1: Default Correspondences

RULE 2: Syllable Boundary Deletion
-:0 <=> _# | V:0 _ [+:0|#] | _ +:0 V:0 | IOU:G_
; *ha-ta-* *po-ass-ta* *ka-ass-ta* *seo-eoss-ta*
; *ha-ta0* *pw0ass-ta* *k00ass-ta* *s0 0eoss-ta*

RULE 3: Vowel Harmony
eo:a => Pv:@ (C:@) -:@ +:0 (0:w) _
; *ka-eo* *po-eo* *top-0eo*
; *k@@a* *p@@a* *to0@wa*

RULE 4: Identical Vowel(*eo*) Adjacency Constraint
eo:eo /<= Pv:@ (C:@) -:@ +:0 (0:w) _
; *seo-eo* suppressed

RULE 5: Stem-Vowel(*a,eo,eu*) Deletion
a:0 => _ -:@ +:0 @:a
; *ka-eo* *seo-eo*
; *k0- a* *s0- eo*

RULE 6: Identical Vowel(*a*) Adjacency Constraint
a:a /<= _ -:@ +:0 @:a
; *ka-a* suppresed

RULE 7: Vowel(*eu*) Deletion
eu:0 => _ -:@ +:0 @:eo
; *theu-eo*
; *th0 -eo*

RULE 8: Identical Vowel(*eu*) Adjacency Constraint
eu:eu /<= _ -:@ +:0 @:eo
; *theu-eo* suppressed

RULE 9: Vowel(*eo*) Deletion
eo:0 => _ -:@ +:0 eo | [e|ae|e:ae] -:@ +:0 _ | h a:ae -:@ +:0 _
; *seo-eo* *kwe -eo* *pe-eo* *nae-eo* *ha -eo*
; *s0 0eo* *kwae00* *pe00* *nae00* *hae00*
; ”eo-eo” contraction is obligatory; the others, optional

RULE 10: Identical Vowel(*a*) Adjacency Constraint
eo:eo /<= _ -:@ +:0 eo

RULE 11: Vowel Shift (*e*→*ae*)
e:ae <=> C w _ -:@ +:0 eo:0
; *kwe -eo*
; *kwae00*

RULE 12-13: Glide Formation
12: Glide Formation Special Correspondences
i o u @
y w w @

13: IOU:G => _ -:@ +:0 V:@
; *ha-si-eo   po-ass-ta   cu-eo*
; *ha-sy0eo pw0ass-ta   cw0eo*

RULE 14-16: Rime "[*u*|*o*]*p*" Irregularity
14: 0:w <=> [u|o] p:@ - +:0 _ eo:@
15: p:0 => [u|o]_ - +:0 0:w eo:@
16: p:p /<= [u|o]_ - +:0 0:w eo:@
; *cup-0eo   top-0a*
; *cu0-weo   to0-wa*

RULE 17-19: *ha-ta* Verb Irregularity
17: a:0 /<= h _ -:@ +:0 eo:@
18: 0:y => h a:@ -:@ +:0 _ eo:@
; *ha-0eo*
; *ha-yeo*
19: a:ae => h _ -:@ +:0 eo:@
; *ha- 0eo*
; *hae-00*

RULE 20: Noun Suffix Sonorant Insertion
0:{n,l} <=> V - +:0 _ eu {n,l} -:0
; *pa-ta-0eun   pa-ta-0eul*
; *pa-ta-neun   pa-ta-leul*

RULE 21: Vowel(*eu*) Insertion
0:eu <=> C - +:0 _ n -:@ #
; *cuk-0 n#*
; *cuk-eun#*

5 Lexicon and Morphotactics

Lexicon not only provides morpheme specific information of glossary or grammatical meanings but also play the role of initial constraints on the morpheme sequences. Each lexical entry is composed of at least three obligatory fields such as ”lexical item”, ”alternation name”, and a ”gloss string”. Let us take a look at the shape of lexical entries.

(12) a. \\lexform *son-*
\\sublexicon N
\\alternation Nfinal
\\gloss Hand

b. \\lf +*ki-*
\\lx SUFFIX
\\alt Suffix
\\fea v/n aj/n
\\gl1 +Nform

c. \\lf 0
\\lx SUFFIX
\\alt End

d. \\lf *hang-sang-*
\\lx AV
\\alt End
\\gl1 Always

The lexical item itself of course is put in the ”lexform” field. Null item ”0” shown in (12c) means suffix affixation is optional so that roots can be realised as words without any suffix attached. In that sense, there should be a null lexical item for prefix as well. The field ”sublexicon” tells which subcategory the item is included in. Usually a predeclared part-of-speech is specified in this field. The ”gloss” field is for any information which will show up when the analysis is performed. I used this this field by filling in a grammitical role if the lexical item is an affix, or a English-translated meaning if the item is a root.

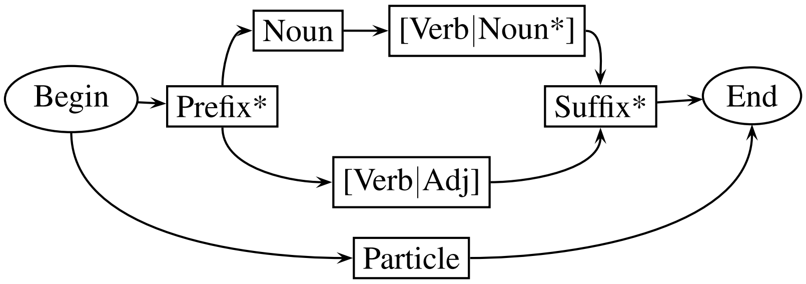
The ”alternation” field is used to constrain the basic sequence of morphemes by restricting the possible morpheme types which may follow. That field of (12a) means N should be followed only by a lexical item whose sublexicon name is one of the elements of the ”Nfinal” group. Of course the ”Nfinal” group members should be specified in advance as in (13) and this is the way Korean noun compounds are established. There are two types of compounding in Korean: first, noun followed by 1 or more nouns, second, noun followed by the so called ”light verb” whose role is to simply provide the semantic feature ”action” to the preceded noun. The latter is a very frequent way of predication in Korean. For instance, a noun *sa-lang* meaning ”love” can be denominalised simply appending a light verb *ha* making a verb stem *sa-lang-ha*. This word formation can easily be handled by the alternation declaration.

In the case of (12c), the ”End” alternation blocks further looping of the suffixation and ends the word formation. As ”AV” in (12d) is one of the elements in ”Particle” alternation in (13), the word *hang-sang* is supposed not to be preceded or followed by any affix. the item cannot be attached by any affixes and looping. (13) shows how the Korean alternation is declared.

(13)	ALTERNATION Particle	IJ AV CJ AN
	ALTERNATION Prefix	PREFIX
	ALTERNATION Root	N V AJ
	ALTERNATION Suffix	SUFFIX
	ALTERNATION Nfinal	N V SUFFIX
	ALTERNATION End	End

According to this declaration Korean initial morphotactics of complex words can be described as Figure 1.

The top path is of course for a noun itself or two ways of noun compounding as explained just above. Roots other than noun (ie, V or AJ) take the other path from prefix node and particles don’t pass through any intermediate nodes.



where "*" means 0 or more occurrences
"|" means any of items within square brackets

Figure 1: Korean Word Structure

For only the analysis of the word structure, more elaborate morphotactic constraints do not have to be established since the input data are supposed to already be well-formed words and some ambiguities can properly be handled by the functions explanined so far. For the synthesising of morphemes into words, however, further morpheme sequence constraints are inevitable to prevent generation of spurious morpheme sequences. For example, consider an ill-formed surface word *son-ki*("Hand"+nominaliser). To properly suppress the occurence of this form we need another constraint saying that *ki* suffix can only follow adjectives or verbs like *ye-ppeu-ki*("Pretty"+nominaliser) or *ha-ki*("Do"+nominaliser) as Figure 1 allowes all the occurences of roots followed by any kind of suffix(es). This job is done by the "feature" field of the lexical entry which is shown in (12a). The features "aj/n" and "v/n" means that the lexical item can only follow an adjective or a verb and the part-of-speech of the form after the affixation is noun. The latter makes it possible that more post-noun suffixes can be further attached producing a correct form like *ha-ki-ka*("Do"+nominaliser+topic_marker).

6 Test and Conclusion

To test the analysis performance, ten Korean newspaper articles were extracted at random from online. A list containing the words made up of two or more morphemes was generated from the texts by simply changing all the spaces into new lines followed by trivial editing.

After the PC-Kimmo is opened, the files of rules, all the sub lexicons, and grammar description are loaded one after another and then the test file or individual words are provided for the analysis. (14) shows an example of an input word and its analysed output.

(14) Analysis Examples

a. Input *ka*
   Output *ka-* (Go)Stem Itself
   ka-+eo- (Go)+Cont/Imp

b. Input *sa-lang-haess-teon*
   Output *sa-lang-ha-+eoss-+teon-*
   (Love)(Do)+Past+Adnom

c. Input *ka-syeoss-kess-cyo*
   Output *ka-+si-+eoss-+kess-+cyo-*
   (Go)+Honor+Past+Guess+RecFin/QueFin

(14a) shows a successful analysis of multiple usages and (14b) reveals how ”light verb” is correctly analysed. In this way, the noun compounds could also be decomposed. Besides, some surface forms which could be derived for only in terms of an extrinsic rule ordering by the rules of generative phonology can now be created easily without any conditioning of the rules thanks to the simultaneous application of rules of two-level rules.

All the corpus internal data are analysed correctly and the data extension along with addition or correction of relevant rules are currently being done.

What should be done in the future is an application of this morphological analysis to more practical speech systems. A letter-to-sound rule for text-to-speech synthesis system of Korean will be made even more plausible if this morphological analysis is performed simultaneously with other post-lexical phonological rules. Speech recognition systems is also expected to have a better shape of dictionary files and subsequently more robust recognition algorithm will be able to be developed.

References

Antworth, Evan L. 1990. *PC-KIMMO: A Two-Level Processor for Morphological Analysis*. Occasional Publications in Academic Computing 16. Texas: Summer Institute of Linguistics.

Black, Alan W. 1990. A computational description of Japanese morphology. Draft.

Chung, Kook. 1994. *Understand Generative Phonology*. Seoul, Korea: Hanshin. (in Korean).

Huh, W. 1991. *Korean Phonology*. Seoul, Korea: Sam Munhwasa. (in Korean).

Karttunen, L. 1983. A two-level morphological analysis of English. *Texas Linguistic Forum* 22: 165–186.

Koskenniemi, K. 1983. Two-level model for morphological analysis. In *IJCAI-83*.

Lee, H. 1996. *Korean Phonetics*. Seoul, Korea: Taehaksa. (in Korean).

Sproat, Richard. 1992. *Morphology and Computation*. ACL-MIT Press Series in Natural Language Processing. MIT Press.