Implementation of Korean Syllable Structures in the Typed Feature Structure Formalism

Gyu-hyung Lee[a], Ye-seul Park[b], and Yong-hun Lee[c]

[a]Department of English Language and Literature, Hannam University,
133 Ojeong-dong, Daedeok-gu, Daejeon 306-791, Korea
gyuhyung73@naver.com
[b]Department of English Language and Literature, Hannam University,
133 Ojeong-dong, Daedeok-gu, Daejeon 306-791, Korea
pys218218@gmail.com
[c]Department of English Language and Literature, Chungnam National University,
220 Gung-dong, Yuseng-gu, Daejeon 305-764, Korea
yleeuiuc@hanmail.net

Abstract. It has been known that the syllable structures in Korean are different from those in English. The goal of this paper is to provide computational implementations for Korean syllable structures in the typed feature structure formalism. The system that we adopted in this paper is the Linguistic Knowledge Building system. We first implemented the type hierarchies and AVMs for *segment* and *suprasegment*. The types *consonant* and *vowel* were included under the type *segment*, and the various different types were included under the type *suprasegment* for syllable structures. Then, we provided the rules for syllable structures. Unlike English syllabification, it has been known that *onset* and *nucleus* form a unit in Korean, which is called *core*. Accordingly, we provided the rules for *onset*, *nucleus*, and *coda*; then, the rules for *core* and *syllable* to combine segments into syllable structures. This paper also employed the type *nf* to solve the ambiguity problems.

Keywords: syllable structure, Korean, typed feature formalism, LKB, implementation

1 Introduction

Since Bird & Klein (1994), there have been many trials to implement phonological processes within the typed feature structure formalism. Most of them have been conducted within Head-driven Phrase Structure Grammar (HPSG; Pollard & Sag, 1994; Sag & Wasow, 1999; Sag et al. 2003).

The goal of this paper is to provide computational implementations for Korean syllable structures within the typed feature structure formalism. The implementational system that we adopted in this paper is the Linguistic Knowledge Building (LKB) system (Copestake, 2002). In this system, we first implemented the type hierarchies for the two types *segment* and *suprasegment*. The types *consonant* and *vowel* were included under the type *segment*. Various different types were included under the type *suprasegment* for syllable structures. Then, we provided rules for syllable structures. It has been known that the syllable structures in Korean are different from those in English. Unlike English syllable structures, it has been known that *onset* and *nucleus* form a *core*, and *core* and *coda* form a *syllable* in Korean. We first provided the implementational rules for *onset*, *nucleus*, and *coda* first; and then we provided rules for *core* and *syllable*. Finally, we used the type *phon-word* in which all the syllables were combined within a phonological word. We also employed the type *nf* in order to solve the ambiguity problems during the parsing processes.

2 Previous Studies

Previous studies related to this paper can be divided into two types. The first group is related to syllable structures, and the other group is on computational phonology in the typed feature structure formalism.

According to Fromkin et al. (2010) and Kenstowicz (1994), English syllable structures are constructed as in Figure 1.

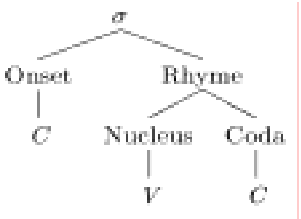

Figure 1: English Syllable Structure

As you can observe in Figure 1, *nucleus* and *coda* form a *rhyme*, and *onset* and *rhyme* form a *syllable*. Several pieces of empirical evidence show that *nucleus* and *coda* form a unit in English.

It has been known that the syllable structures in Korean are different from those in English. There have been two types of analyses on Korean syllable structures, which are graphically illustrated in Figure 2. Here, C and V refer to consonant and vowel respectively, and G refers to glide. Because glides are also consonants, the symbol C in this paper refers to all the other consonants except glides.

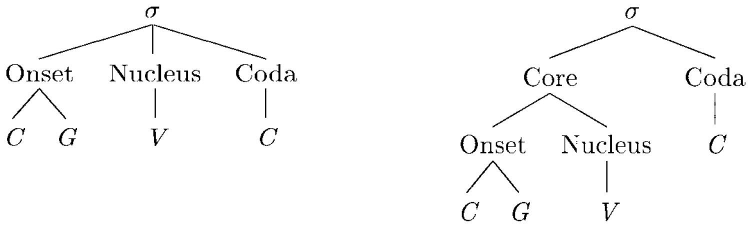

Figure 2: Korean Syllable Structure (Ternary Branching (Left) and Binary Branching (Right))

The ternary branching analyses, which were included in Lee (1982), Ahn (1985), and Park (1993), said that there is no internal structure within the syllable and that a *syllable* is composed of just three components *onset*, *nucleus*, and *coda*. The binary branching analyses, which were included in Kim (1987), Park (1985), Jun (1980), Ahn (1988), and Kim (1989), claimed that *onset* and *nucleus* form a unit which is called *core*, and that *core* and *coda* form a *syllable*. Many previous studies including Kang (2003) showed that the second type of analyses is suitable for Korean syllable structures, and this paper also adopted the binary branching analysis in the computational implementation.[1] In the above analyses, note that the template for Korean syllabification is CGVC, whether the syllable has binary branching or ternary branching.

There have been many trials to implement phonological processes within the typed feature structure formalism. Bird and Klein (1994) was virtually the first paper which tried to implement phonological analyses in typed feature systems. Following this paper, Orgum (1996) developed sign-based morphology and phonology, which can easily be implemented in the

[1] As one of the evidences of Binary Branching syllable structures in Korean, Kang (2003) and many other scholars mentioned partial reduplication phenomena in Korean where 'Onset + Nucleus', rather than 'Nucleus + Coda' is reduplicated.

HPSG framework. Recently, Tseng (2008) tried to implement the syllable structures in western languages such as English, and Skwarski (2009) included underlying and surface forms in the analyses.

3 Korean Syllable Structures in Typed Feature Formalism

3.1 Type Hierarchy and AVMs

The first step to provide computational implementations of Korean syllable structures in the LKB system starts from the type hierarchies in which major concepts of syllable structures are organized. Figure 1 illustrates the type hierarchies for *segment* and *suprasegment*.

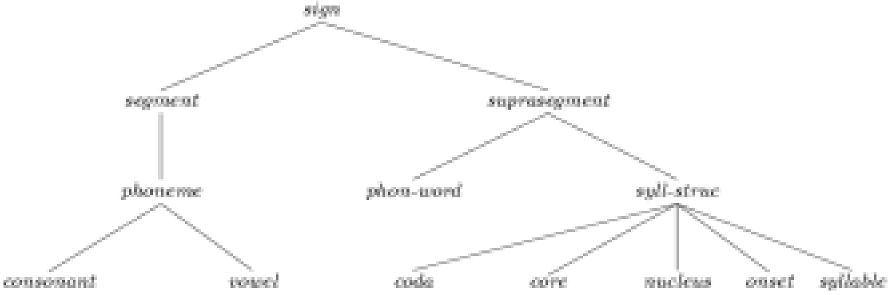

Figure 3: Type Hierarchy for *segment* and *suprasegment*

As you can find in Figure 1, *sign* is divided into *segment* and *suprasegment*, which are the types for segmental and suprasegmental elements respectively. The type *segment* contains the type *phoneme*, and it is subdivided into two sub-types *consonant* and *vowel*.[2] The type *suprasegment* contains *syll-struct*, and all the types which are related to syllable structures are contained here. Also, note that *phon-word* (phonological word) is included under the type *suprasegment*.

The type *phoneme* contains two sub-types *consonant* and *vowel*, and their attribute-value matrixes (AVM) are shown in Figure 4. Since both *consonant* and *vowel* are subtypes of *phoneme*, they inherit the properties of *phoneme*.

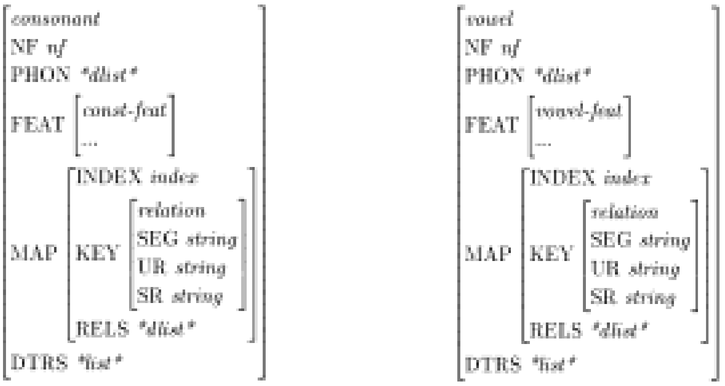

Figure 4: AVMs for *consonant* and *vowel*

[2] It does not mean that the type *segment* contains only *phoneme*. It also contains a few more types such as *allophone*, under which various kinds of allophones are enumerated. Because this paper is about syllable structures, other sorts of phonological types are omitted here. The same reasoning also holds for the type *suprasegment*. A few more types are also included under the type *suprasegment*, but they are not shown here for convenience.

NF is the attribute for *normal form*, and its main contribution to the system is to reduce the ambiguity during the parsing processes.[3] PHON is the attribute for phonetic forms of each segment, and difference lists (**dlist**) are used here. FEAT contains consonant features (*const-feat*) and vowel features (*vowel-feat*). MAP is used to map the underlying representations (UR) to the surface representations (SR).[4] DTRS is the attribute for daughters.

The type *suprasegment* also has the AVM, and Figure 5 shows us the structure of the AVM.

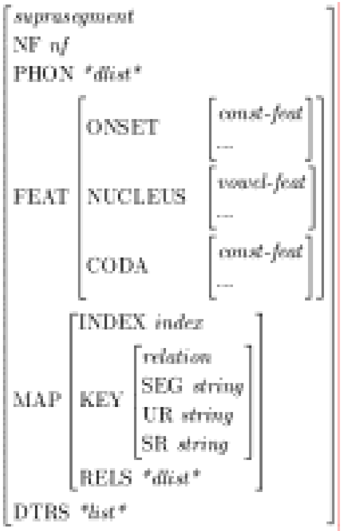

Figure 5: AVM for *suprasegment*

As you can observe, the AVM for *suprasegment* is similar to that for *consonant* and *vowel*. The only difference is that the attribute FEAT is further sub-divided into three attributes ONSET, NUCLEUS, and CODA. Because *onset* and *coda* must be a *consonant*, the AVMs for ONSET and CODA contain *const-feat*. On the other hand, since *nucleus* must contain a *vowel*, the AVM for NUCLEUS contains *vowel-feat*. Everything else is identical with the AVMs for *consonant* and *vowel*.

3.2 Rules for Syllable Structures

Now that we have the type hierarchies and AVMs for *consonant*, *vowel*, and *suprasegment*, it is the time to implement the rules for syllable structures. As mentioned in Section 2, this paper adopts the binary branching analysis in the computational implementations. That is, *onset* and *nucleus* form a *core* and that this *core* combines with *coda* in order to form a *syllable* in our implementations. Therefore, the rules for these types are necessary.

The first kind of rule is those for *onset*, *nucleus*, and *coda*. The function of these rules is to connect each segment (*consonant* and *vowel*) to the types for syllable structure (*syll-struc*). The rules are schematically illustrated in Figure 6.[5,6,7]

[3] Section 3.3 includes more detailed explanations about this attribute.

[4] The ideas which are enumerated in Skwarski (2009) are partially implemented in the AVMs for MAP.

[5] You may find that one more onset rule is necessary. In the right part of Figure 2, note that glide can go between *consonant* and *vowel*. In these cases, the following three values of *onset* are constructed by concatenating the corresponding values of the two constituents *consonant* and *glide*: (i) the PHON value, (ii) the FEAT|ONSET value, and (iii) the MAP|RELS value. As in footnote 1, *consonant* refers to all the other consonants except glides. The concatenating operation can be represented by ⊕ as in Figure 7 and Figure 8.

[6] As pointed out in the footnote 9, this paper used Romanization system for the orthographies of the Korean language in the actual implementation. Since glides are included in the orthographies of vowel, like 'ya' or 'yu', this paper does not contain the detailed discussions on the characteristics of glides.

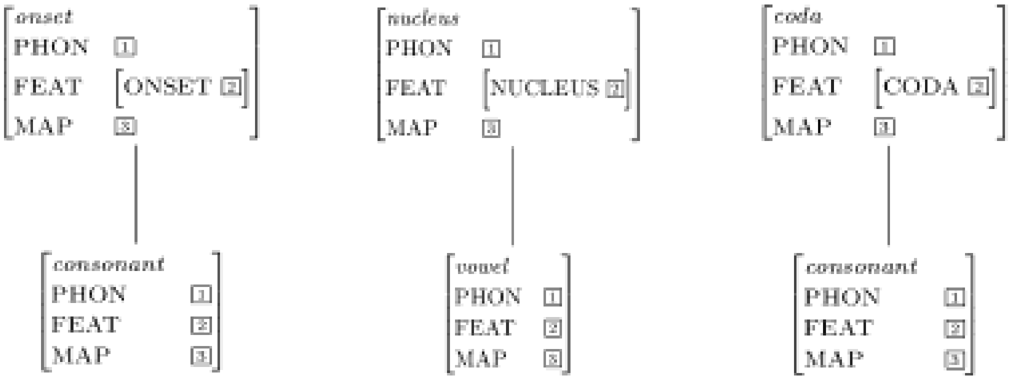

Figure 6: Onset Rule (Left), Nucleus Rule (Middle), and Coda Rule (Right)

As you can see in these rules, *consonant* can be either an *onset* or a *coda*, and the FEAT value of *consonant* is percolated up into the FEAT|ONSET value and/or the FEAT|CODA value. Likewise, since *vowel* can be a *nucleus*, the FEAT value of *vowel* is reflected in the FEAT| NUCLEUS value.

Now, it is time to provide a rule for *core*. As you can observe in Figure 2, the main function of *core* is to combine two constituents, *onset* and *nucleus*. There are two sorts of *core* structure. One is the structure where *core* includes only one type *nucleus*, and the other is the one which is composed of two constituents, *onset* and *nucleus*. Therefore, two kinds of rules have to be provided separately for the type *core*. Figure 7 illustrates these two rules.

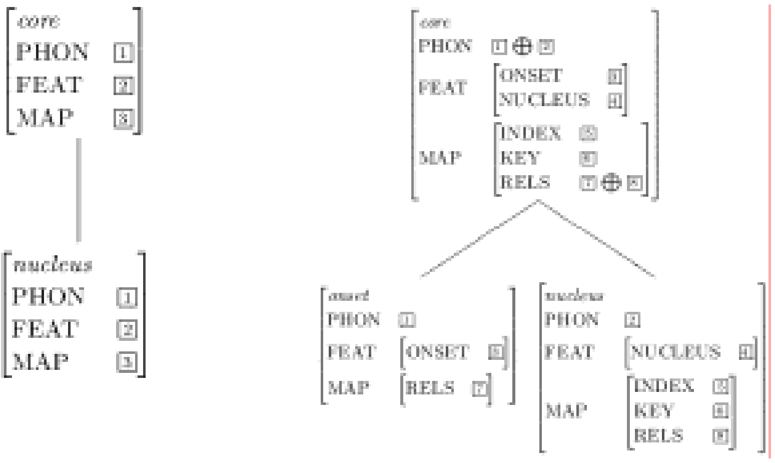

Figure 7: Core Rules

When a *core* includes only *nucleus*, what we have to do is to percolate up the values of each attribute. The left *core* rule is for this case. When a *core* is composed of *onset* and *nucleus*, the features of *onset* and *nucleus* are unified. The PHON values of *onset* and *nucleus* are concatenated. The FEAT|ONSET value of *onset* is percolated up into the FEAT|ONSET value of *core*, and the FEAT|NUCLEUS value of *nucleus* is percolated up into the FEAT|NUCLEUS value of *core*. When *onset* and *nucleus* combine to form a *core*, since *nucleus* is the most important (head) of the *syllable*, the MAP|INDEX value and the MAP|KEY value of *nucleus* are

[7] A reviewer pointed out that the rules for *onset* and *coda* potentially introduce ambiguity and asked how this problem can be solved. This problem can be handled with the phonotactic constraints, which were not described in this paper, because of the limit of space.

percolated up into the corresponding values of *core*. The right *core* rule represents these mechanisms.

Finally, we are ready to provide a rule for *syllable*. As you can observe the binary branching analysis in Figure 2, a *syllable* is composed of two constituents, *core* and *coda*. As in the type *core*, there are two kinds of *syllable* structure. One is the structure where *syllable* includes only *core* (the case where syllable contains no *coda*), and the other is the structure which is composed of two constituents, *core* and *coda*. Therefore, two different types of rules have to be made separately for the type *syllable*. Figure 8 illustrates these two rules.

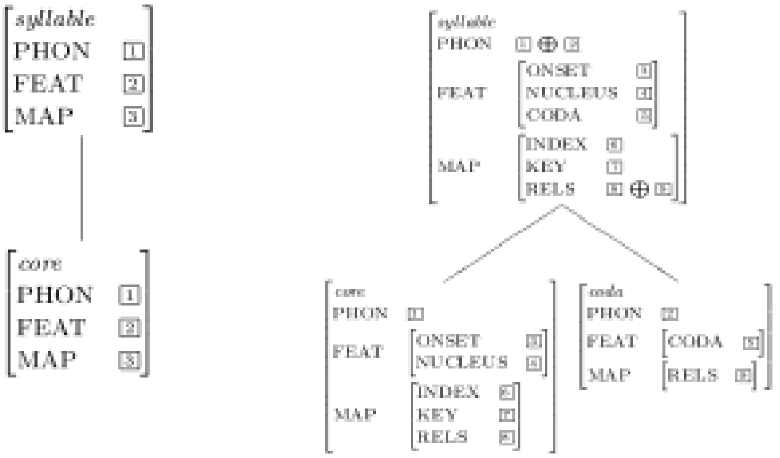

Figure 8: Syllable Rules

When a *syllable* includes only *core*, what we have to do is to percolate up the values of each attribute. The left *syllable* rule is used for this end. When a *syllable* is composed of two components, *core* and *coda*, the features of *core* and *coda* have to be unified. The PHON values of *core* and *coda* are concatenated. The FEAT|ONSET value and the FEAT|NUCLEUS value of *core* are percolated up into the corresponding values of *syllable*, and the FEAT|CODA value of *coda* is percolated up into the FEAT|CODA value of *syllable*.[8] When *core* and *coda* combine to form a *syllable*, since *core* contains the features for *nucleus* (the head of a *syllable*), the MAP| INDEX value and the MAP|KEY value of the type *core* are percolated up into the corresponding values of *syllable*. The right *syllable* rule is for these cases.

Though this paper does not include the rules for *phon-word*, there are also rules for *phon-word*. The main function of these rules is to combine the syllables to form a (phonological) word.

3.3 Resolving Ambiguity Problems

The last part of this section is devoted to the ambiguity problems, i.e. how to handle the ambiguity problems during the parsing processes.

When two consonants go between the vowels, we have no problem. For example, if we have a word '*kwukmin* [kukmin]' whose meaning is 'the people of a nation', two consonants 'k' and 'm' go between the vowels 'u' and 'i'.[9] In this case, the first consonant 'k' is syllabified into the

[8] Although how the FEAT values are used is not explained in detail here, these attributes are made for the purpose of modeling many phonological phenomena. For example, when we try to implement *assimilation* rules, the rules have to confer the phonological features of each segment. The FEAT values are used in those kinds of cases. Since this paper is implementing only syllable structures, the usages and detailed explanations for these values are not included here. However, they play crucial roles in the system, when we model the phonological phenomena such as *assimilation*.

[9] Here, Yale Romanization system is used for the orthography, and IPA symbols are used for the pronunciation. IPA symbols are indicated by brackets, and they are used as an input string in the actual implementation in Section 4.

coda position of the first syllable, and the second consonant ‘m’ is syllabified into the onset position of the second syllable. Problems occur when only one consonant goes between two syllables. For example, in case of a word ‘*nala* [nala]’ whose meaning is ‘a nation’, two possible structures in Figure 9 are generated to this word.

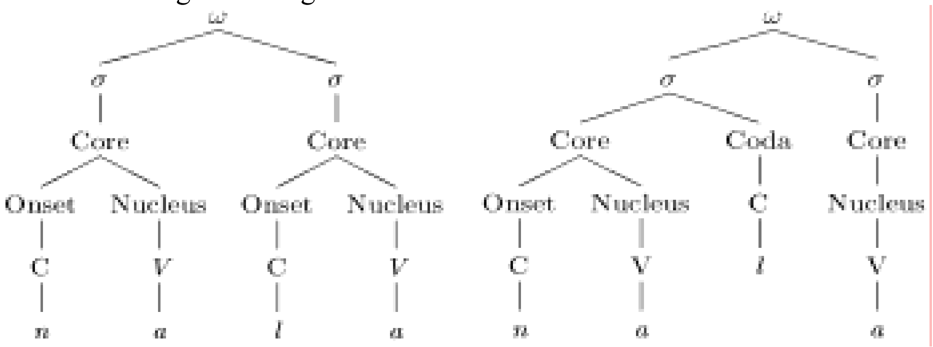

Figure 9: Two Possible Syllable Structures

In the first syllable structure, the second consonant ‘l’ is syllabified into the onset position of the second syllable. In the second syllable structure, however, it is syllabified into the coda position of the first syllable. Since a consonant can be syllabified either into an onset position or a coda position, this kind of ambiguity may occur during the parsing processes. In the actual syllabification, however, the left syllable structure is correct for the word ‘*nala*’. Therefore, it is necessary to make the right syllable structure not generated during the parsing processes.

In order to resolve this problem, we employed the type *nf* (Normal Form), and its type hierarchy is shown in Figure 10.

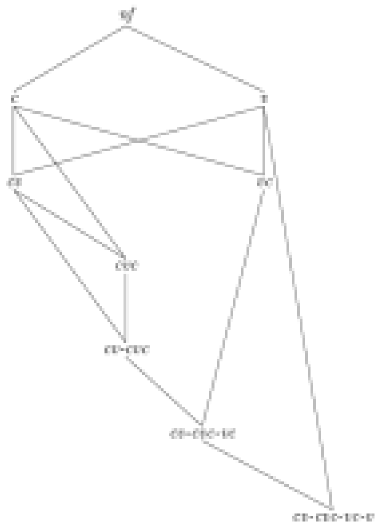

Figure 10: Type Hierarchy for *nf*

Beavers (2002, 2004) also used the type *nf* in his Type-inherited Combinatory Categorial Grammar (TCCG). The purpose of using the type *nf* in the TCCG system is identical with that of this paper: to reduce the syntactic ambiguity during the parsing processes. Accordingly, we also adopted this type to solve the ambiguity problems.

At the bottom of the type hierarchy in Figure 10, you may find the type *cv-cvc-vc-v*. It represents the preference ranking of the syllable structures in Korean. According to this ranking, the CV structure is the most preferable one, and CVC follows the CV structure. Then VC follows CVC, and V is the least preferable one in Korean.

As you can observe in Figure 4 and Figure 5, these preference rankings are encoded in the NF value of *segment* and *suprasegment*. Since *consonant* and *vowel* are sub-types of *segment* and *syll-struct* is a subtype of *suprasegment*, all the consonants and vowels and all the types under *syll-struct* have the value for *nf*.

Here, note that the CV structure is more preferable than the CVC one. Now, compare the two syllable structures in Figure 9, those for '*nala*'. The left syllable structure has a form 'CV+CV', whereas the right syllable one has a form 'CVC+V'. Because the CV structure is more preferable than the CVC one, the left syllable structure is preferred to the right syllable one. Therefore, the parser correctly selects the left syllable structure for the word '*nala*'.

4 The LKB Implementation Examples

In this section, some actual implementation examples are shown. Three words are selected for the illustration. The first one is '*a.i* [ai]' whose meaning is 'a child or children', the second one is '*nala* [nala]' whose meaning is 'a nation', and the last one is '*kwukmin* [kukmin]' whose meaning is 'the people of a nation'.

Figure 11 illustrates the analysis result of the first word '*a.i* [ai]'.

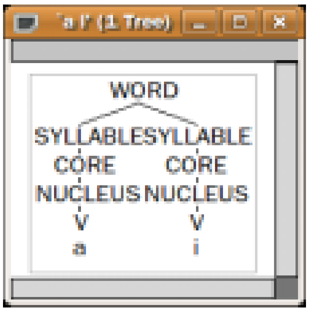

Figure 11: Implemented Example 1

Since this word contains only vowels, the first rules for *core* and *syllable* are applied. That is, *core* contains only *nucleus*, and *syllable* contains only *core*. Therefore, the syllable structure for this word is as shown in Figure 11.

Figure 12 illustrates the analysis result of the second word '*nala* [nala]'.

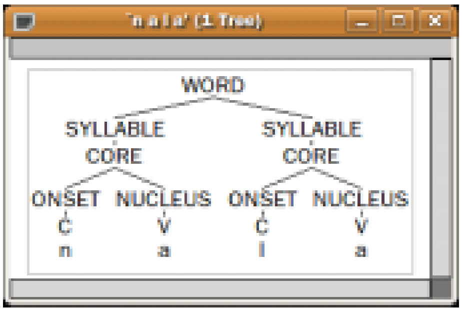

Figure 12: Implemented Example 2

Since this word has a syllable structure 'CV+CV', the second *core* rule and the first *syllable* rule are applied. Accordingly, a *core* contains both *onset* and *nucleus*, but *syllable* contains only *core*. Therefore, the syllable structure for this word is as shown in Figure 12.

Figure 13 illustrates the analysis result of the third word ‘*kwukmin* [kukmin]’.

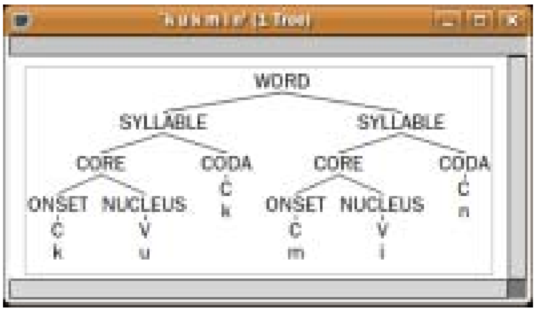

Figure 13: Implemented Example 3

Since this word has a syllable structure ‘CVC+CVC’, the second rules are applied for *core* and *syllable.* That is, *core* contains both *onset* and *nucleus*, and *syllable* contains only both *core* and *coda*. Therefore, the syllable structure for this word is as shown in Figure 13.

Through the computational implementations, we could find that the system developed in this paper correctly analyzed the syllable structures in Korean.[10]

5 Conclusion

In this paper, we tried to provide computational implementations for Korean syllable structures within the typed feature structure formalism. For the computational implementation, we adopted the LKB system. In this system, we first implemented the type hierarchies for *segment* and *suprasegment*. The type *segment* contains *consonant* and *vowel*, and various types for syllable structures were included under the type *suprasegment*. Then, we provided various rules for syllable structures. Unlike English syllabification, *onset* and *nucleus* form a *core*, and *core* and *coda* form a *syllable* in Korean. We provided the implementational rules for *onset*, *nucleus*, and *coda*; then, we provided rules for *core* and *syllable*. Finally, we provided the rules for *phon-word* in which syllables were combined. Finally, we also employed the type *nf* in order to solve the ambiguity problems during the parsing processes.

The system developed in this paper can be extended further to include some phonological theories such as feature geometry and underspecification. The system can also be extended to model some phonological phenomena such as assimilation or vowel harmony.

References

Ahn, Sang-Cheol. 1985. *The Interplay of Phonology and Morphology in Korean*. Ph.D. Dissertation. University of Illinois at Urbana-Champaign.

Ahn, Sang-Cheol. 1988. A Revised Theory of Syllable Phonology. *Linguistic Journal of Korean* 13.2: 333-362.

Beavers, J. 2002. A CCG Implementation for the LKB. LinGO Working Paper #2002-8. Stanford, CA: CSLI.

Beavers, J. 2004. Type-inherited Combinatory Categorial Grammar. In *Proceedings of the 20th International Conference on Computational Linguistics*. Article No. 57.

Bird, S. and E. Klein. 1994. Phonological Analysis in Typed Feature Systems. *Computational Linguistics* 20.3: 455-491.

[10] A reviewer pointed out that there was no evaluation reported in the paper. We are now developing an evaluation procedure for this system, and the results come from running the implementation over a large scale of corpus data.

Copestake, A. 2002. *Implementing Typed Feature Structure Grammar*. Stanford, CA: CSLI.

Fromkin, V., R. Rodman, and N. Hymas. 2010. *Introduction to Language*. York: Wadsworth.

Jun, Sang-Beom. 1980. Phonological Interpretation of Lapsus Linguae. *Korean Journal of Linguistics* 5.2:15-32.

Kang, Ongmi. 2003. *Korean Phonology*. Soeul: Taehaksa.

Kenstowicz, M. 1994. *Phonology in Generative Grammar*. Oxford: Blackwell.

Kim, Cha-Kyun. 1987. A Study on Syllable Structure and Some Processes in Its Nucleus in Korean. *Mal* (Language) 12:25-69.

Kim, Jong-Hoon. 1989. English Syllable Structure and Phonological Rules in Korean. *Journal of English Language and Literature* 5.3:589-608.

Lee, Byung-Gun. 1982. A Well-formedness Condition on Syllable Structure. In the Linguistic Society of Korea, (ed.), *Linguistics in the Morning Calm* I, 489-506. Seoul: Hanshin Publishing Co.

Orgum, C. 1996. *Sign-based Morphology and Phonology with Special Attention to Optimality Theory*. Ph.D. Dissertation. University of California at Berkeley.

Park, Chang-Won. 1993. Morphology and Phonology of Onomatopoenia. *Saykukesaynghwal* 3.2:16-52.

Park, Jong-Hee. 1985. On Non-vocalization in Korean. *Journal of Korean Linguistics* 14:189-214

Pollard, C. and I. Sag. 1994. *Head-driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Sag, I. and T. Wasow. 1999. *Syntactic Theory*: *A Formal Introduction*. First Edition. Stanford, CA: CSLI.

Sag, I., T. Wasow, and E. Bender. 2003. *Syntactic Theory*: *A Formal Introduction*. Second Edition. Stanford, CA: CSLI.

Skwarski, F. 2009. Accounting for Underlying Forms in HPSG Phonology. *The Proceedings of the 16th International Conference on Head-Driven Phrase Structure Grammar*, 318-337.

Tseng, J. 2008. The Representation of Syllable Structures in HPSG. *The Proceedings of the 15th International Conference on Head-Driven Phrase Structure Grammar*, 234-252.