LIN 667 Final Project

Soo Hyun Ryu

December 18, 2018

1 Introduction

This project aims to build a system that can represent Korean linguistic knowledge using a constraint-based grammar called Head-Driven Phrase Structure Grammar. Even though there has been a project (Kim & Yang, 2003) to implement Korean Phrase Structure Grammar(KPSG) into Linguistic Knowledge Building (LKB) system, which is Prolog-based grammar and lexicon development environment, the present project has some different points. First, the previous system requires all lexicon items to be stored in the system. Also, Hangul, the original Korean alphabet is not able to be used as input of the system. To suggest a different approach, the current project project integrates KoNlpy (Park & Cho, 2014), a Python package for natural language processing of the Korean language so that Hangul can be dealt properly in the system and that the need for the all lexical entries to be stored is avoided. The only lexical information this project includes is the valency information of the verbs extracted from the previous LKB system(Kim & Yang, 2003).

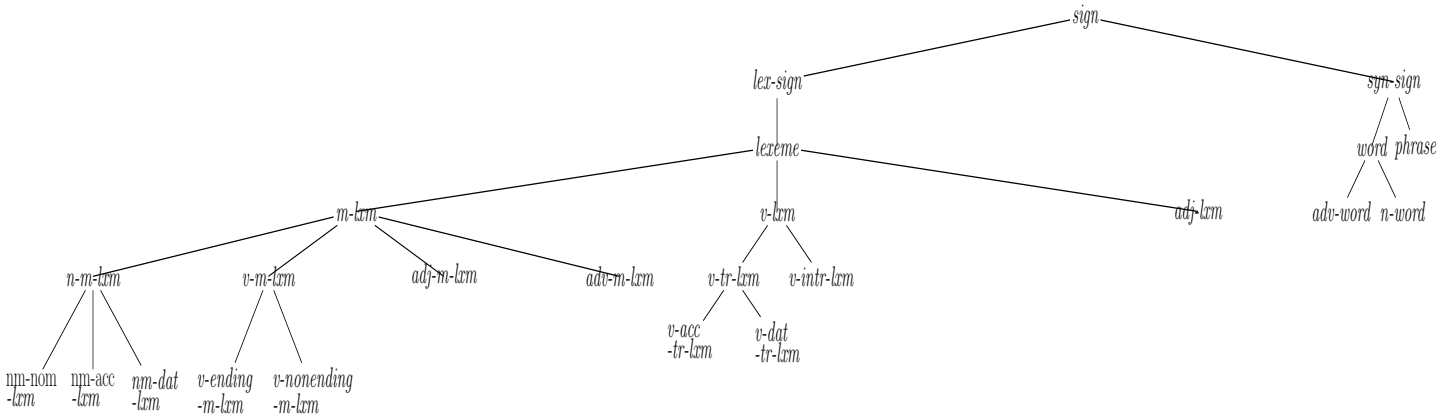
2 Korean Phrase Structure Grammar

Korean grammar is an agglutinative language, in which affixes are added to words in order to decide the words' meanings and grammatical functions. This inflectional system makes Korean grammar complex, but remarkable systematicity of the inflectional system in Korean allows the productivity of the language.

2.1 Type hierarchy

The types of Korean linguistic information in this simple system is classified as Figure 1. While elements in *syn-sign* can directly be inputs in syntax, elements in *lex-sign* need to be combined with another elements to be inputs in syntax according to lexical rules, which will be explained further in the next section. Even though there should be more types of lexemes to deal with more complicated syntax in Korean, this project introduces three types of lexemes: i) *m-lxm* (*'m'* refers to *'marker'*), ii) *v-lxm* and iii) *adj-lxm*.

Elements in *v-lxm* are later function as verbs in the sentences after being combined with appropriate marker lexeme. They are subdivided into *v-tr-lxm* and *v-intr-lxm* depending on valence it takes. (Ditransitive verbs are not included in this project.) *v-tr-lxm* consists of two types, one of which is *v-acc-tr-lxm* taking accusative noun as complement and the other of which is *v-dat-tr-lxm* taking dative noun as complement.



Elements in *adj-lxm* are later function as adjectives or adverbs after being combined with appropriate marker lexemes.

Elements in *m-lxm* function as markers of lexical items, specifying the roles of units that they combine with. *m-lxm* is further subdivided into *n-m-lxm*, *v-m-lxm*, *adj-m-lxm* and *adv-m-lxm*, which function as markers of nouns, verbs, adjectives and adverbs respectively. *n-m-lxm* is further divided into *nm-nom-lxm*, *nm-acc-lxm*, and *nm-dative-lxm*, each one marks noun as nominative, accusative and dative. *v-m-lxm* is divided into *v-ending-m-lxm* and *v-nonending-m-lxm*. While elements in *v-ending-m-lxm* only serve a grammatical function of converting *v-lxm* into a complete input unit in syntax, elements in *v-nonending-m-lxm* enrich the verbal unit by adding a variety information such as passivity, honorific, tense or mood.

Execution example

In the constraint of *sign*, which is the most general constraint of Korean lexical items, has two subfeatures in its 'SYN' feature, which are 'head' and 'valence'. 'Head' includes 'pos' features that indicate the part-of-speech of the lexical item ,even though *sign*'s subclasses might include more features in 'head' feature. In 'valence' feature, there are four subfeatures: i) mod, ii) marking, iii) comps and iv) spr. The 'mod' feature indicates the constraint of the lexical entry that the corresponding lexical entry modifies. The 'marking' feature indicates the constraint of the lexical entry that has to mark the corresponding lexical entry so that it can be converted into *word* or *phrase* which can be input of the syntax.

In the constraint of *m-lxm*, there is a feature of 'mark' in head feature. This gives information of which lexical item it marks. Therefore, in the constraint of *n-m-lxm*, the mark feature is 'noun' while it is 'verb' in *v-m-lxm*.

```
[ orth = ?i                                ]
[                                          ]
[        [ head = [ pos = ?a ]     ] ]
[        [                         ] ]
[ syn  = [        [ comps   = ?e ] ] ]
[        [ val  = [ marking = ?f ] ] ]
[        [        [ mod     = ?g ] ] ]
[        [        [ spr     = ?d ] ] ]
[                                          ]
[ type = ?h                                ]
```

Figure 1: Constraint of SIGN

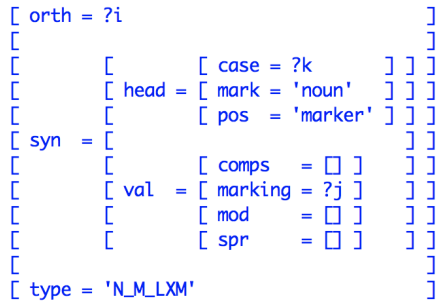


Figure 2: Constraint of n-m-lxm

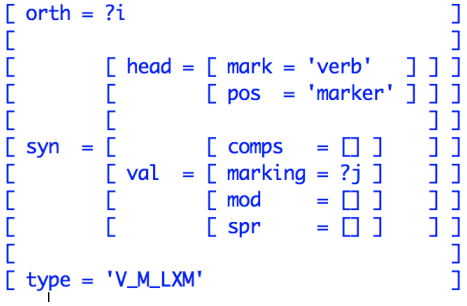


Figure 3: Constraint of v-m-lxm

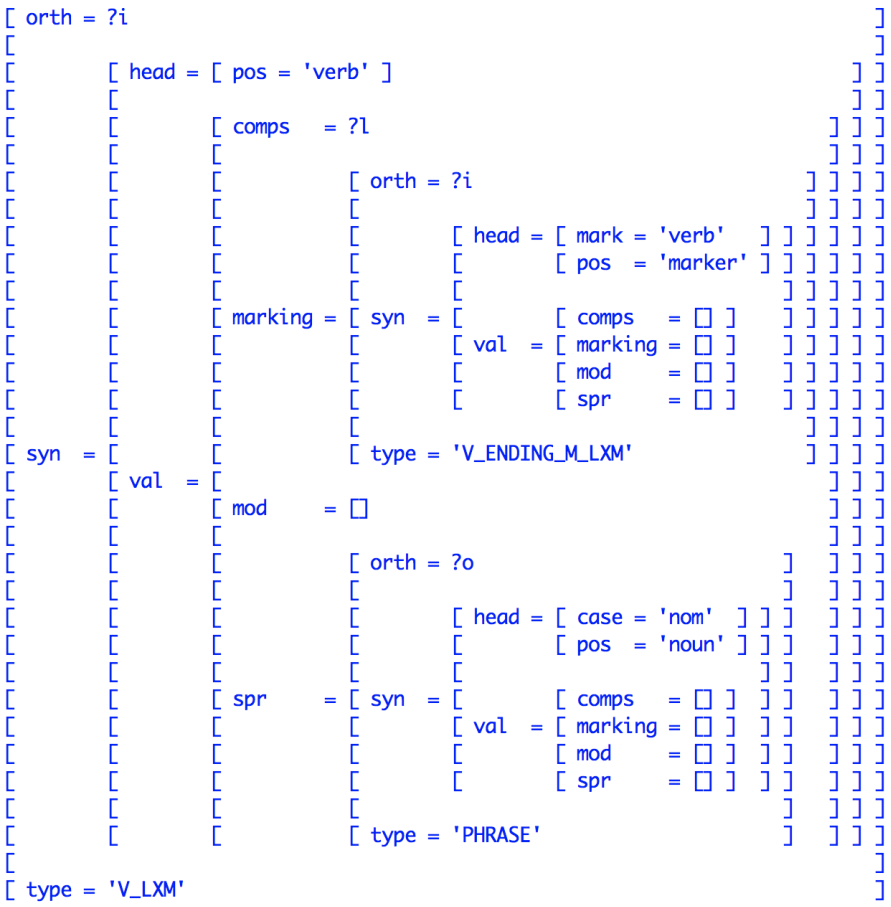


Figure 4: Constraint of v-lxm

2.2 Lexical Rules

Lexical rules in this grammar allows lexemes to be converted into a unit that can be input of the syntax by combining with appropriate lexemes. In this project, four types of lexical rules are implemented: i) nominal lexical rules, ii) verbal lexical rules, iii) adjective lexical rules iv) adverbial lexical rules.

2.2.1 Nominal

Nominal lexical rules combines the *n-word* and *n-m-lxm* in order to generate noun phrases which is attached with its grammatical function, such as nominative, accusative or dative. The reason why *n-word* is not the subclass of *lexeme* is because sometimes nouns are not required to be combined with marker. In other words, elements in *n-word* can be used as input of syntax without the specification of *n-m-lxm*. (The current system requires nouns to be combined with marker.)

Execution example

a. 강아지-에게 'to dog'

dog-DAT

```
>>> cat('강아지')
[ orth = '강아지'                                                    ]
[                                                                    ]
[       [ head = [ case = ?b     ]                                   ] ]
[       [        [ pos  = 'noun' ]                                   ] ]
[       [                                                            ] ]
[       [       [ comps  = □ ]                                     ] ] ]
[       [       [                                                  ] ] ]
[       [       [          [ orth = ?i                           ] ] ] ]
[       [       [          [                                     ] ] ] ]
[       [       [          [       [       [ case = ?k       ] ] ] ] ] ]
[       [       [          [       [ head = [ mark = 'noun'  ] ] ] ] ] ]
[ syn  = [      [          [       [        [ pos  = 'marker' ] ] ] ] ] ]
[       [       [ marking = [ syn  = [                           ] ] ] ] ]
[       [ val  = [          [       [       [ comps  = □ ]     ] ] ] ] ]
[       [       [          [       [ val  = [ marking = ?j ]   ] ] ] ] ]
[       [       [          [       [        [ mod     = □ ]   ] ] ] ] ]
[       [       [          [       [        [ spr     = □ ]   ] ] ] ] ]
[       [       [          [                                     ] ] ] ]
[       [       [          [ type = 'N_M_LXM'                    ] ] ] ]
[       [       [                                                  ] ] ]
[       [       [ mod     = □                                      ] ] ]
[       [       [ spr     = □                                      ] ] ]
[                                                                      ]
[ type = 'N_WORD'                                                      ]
```

Figure 5: Constraint of n-word 'dog'

```
>>> cat('강아지에게')
[ orth = '강아지,에게'                                  ]
[                                                      ]
[       [ head = [ case = 'dat'  ] ] ]
[       [        [ pos  = 'noun' ] ] ]
[       [                          ] ]
[ syn  = [       [ comps  = □ ]  ] ]
[       [ val  = [ marking = □ ]  ] ]
[       [        [ mod     = □ ]  ] ]
[       [        [ spr     = □ ]  ] ]
[                                  ]
[ type = 'PHRASE'                  ]
```

Figure 6: Constraint of dative noun phrase

2.2.2 Verbal

Verbal lexical rules combines the *v-lxm* and *v-m-lxm* in order to generate verb words to be used as input of syntax. To fully cover the productivity of verbal inflection system in Korean, three different verbal lexical rules are implemented. The first rule simply combine the element from *v-lxm* and one from *v-ending-m-lxm*. The second rule combine two different elements from *v-non-ending-m-lxm* to generate another *v-non-ending-m-lxm* which includes information from both elements. The last rule combine *v-non-ending-m-lxm* and *v-ending-lxm* to generate another *v-ending-m-lxm* which includes infromationa from both elements. With these three rules, the semantics of verb words can be fully enriched by being added with multiple markers.

Execution example

a. 잤-다(자-었-다) 'slept'

Sleep-Past-Declaration

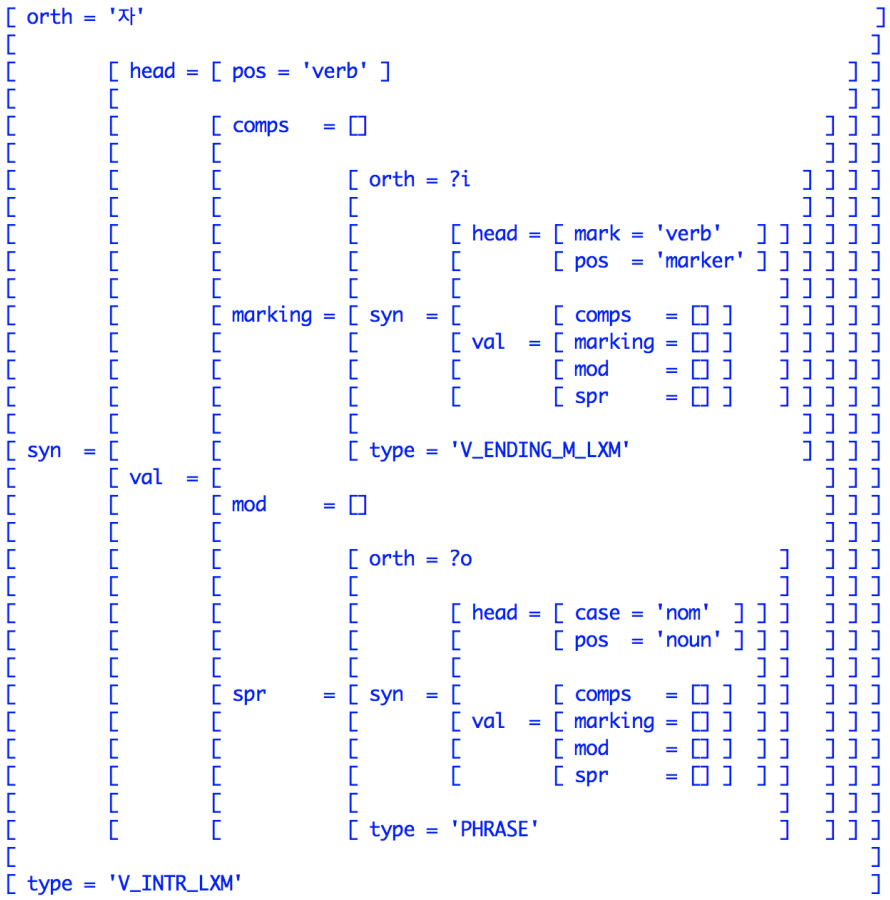


Figure 7: Constraint of *v-lxm* 'sleep'

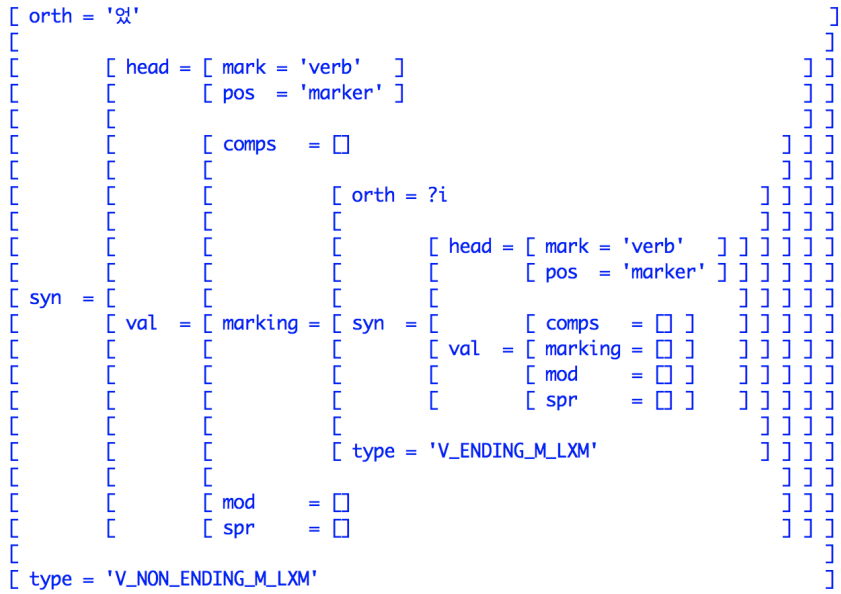


Figure 8: Constraint of *v-non-ening-m-lxm* 'past'

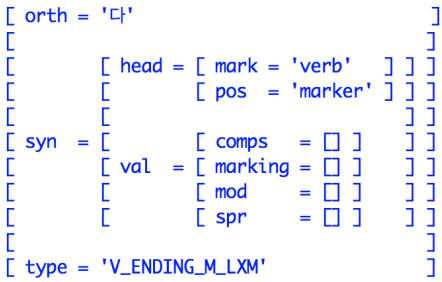


Figure 9: Constraint of *v-ening-m-lxm*

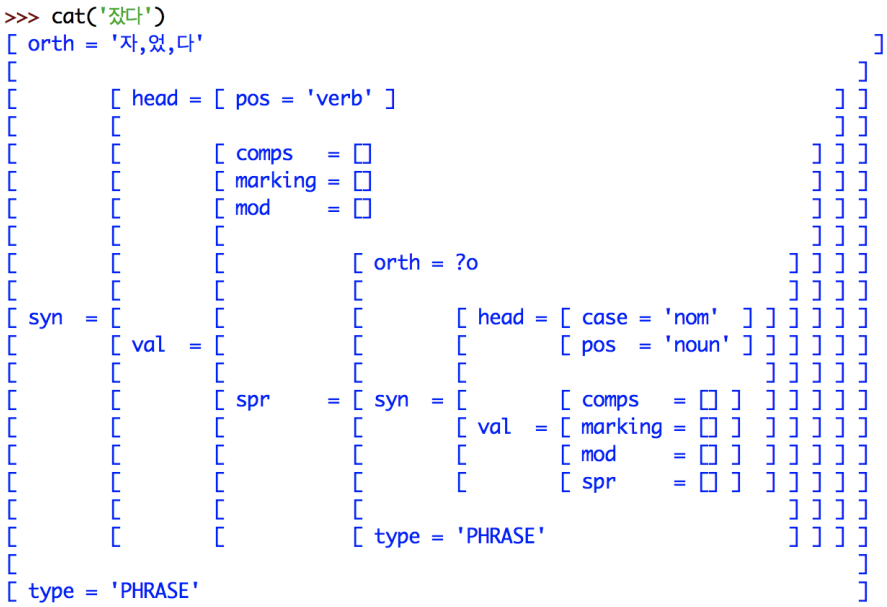


Figure 10: Constraint of v-phrase 'slept'

2.2.3 Adjective

Adjective lexical rules combines the *adj-lxm* and *adj-m-lxm* in order to generate adjective words.

Execution example

a. 좋-은 'nice'

Nice-Adj

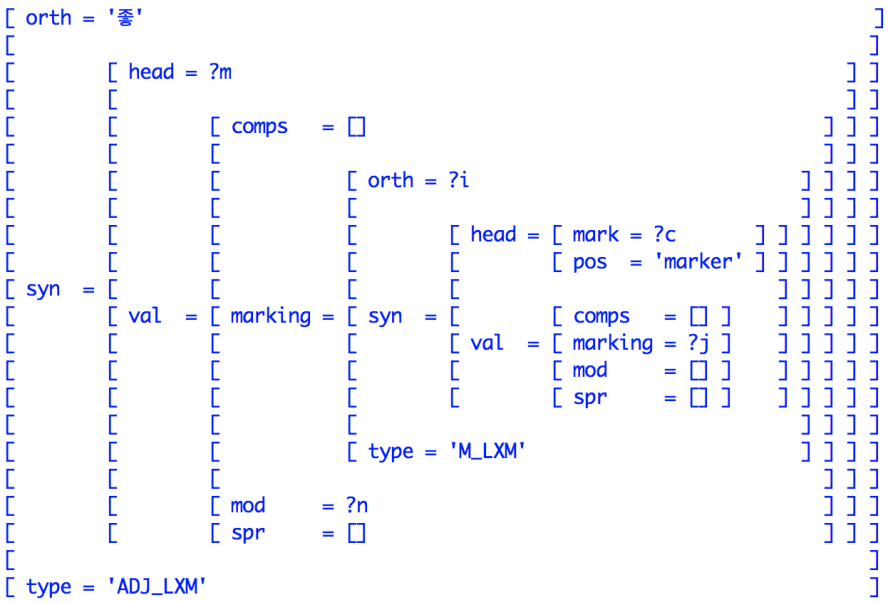


Figure 11: Constraint of *adj-lxm* 'nice'

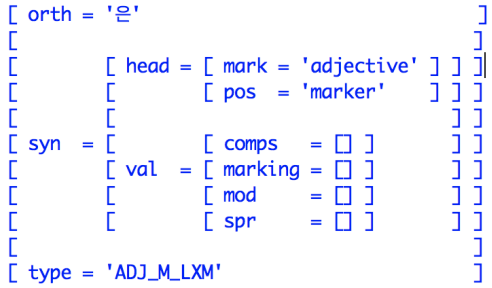


Figure 12: Constraint of *adj-m-lxm*

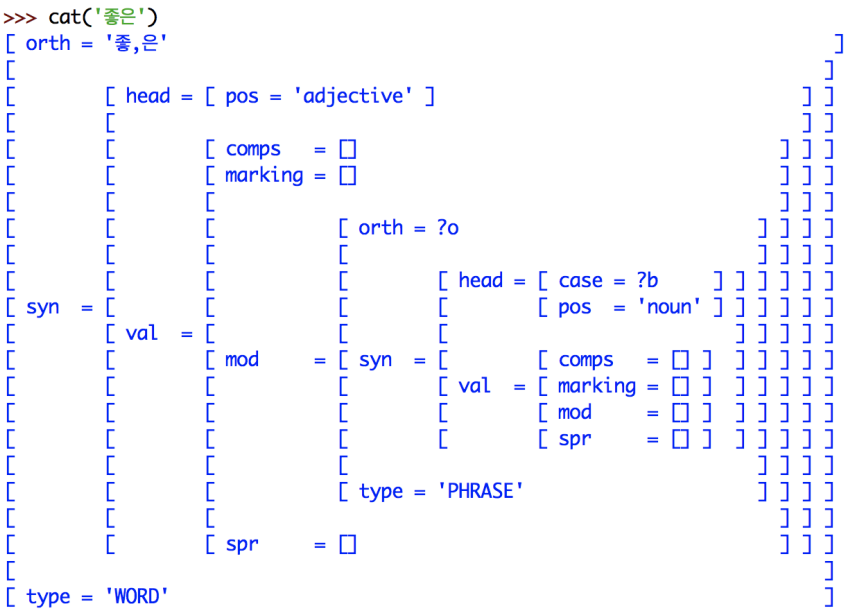


Figure 13: Constraint of Adjective 'nice'

2.2.4 Adverbial

Adverbs in Korean is generated in two ways: while some adverb words exist without requiring any lexical rules, others are generated with combining *adj-lxm* and *adv-m-lxm*. In order to cover the latter case of the adverbs, adverbial lexical rule is implemented in this project, which combines the *adj-lxm* and *adv-m-lxm* in order to generate adverb words.

Execution example

a. 좋-게 'well'

Nice-Adv

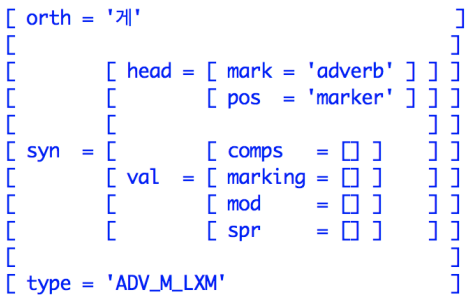


Figure 14: Constraint of *adv-m-lxm*

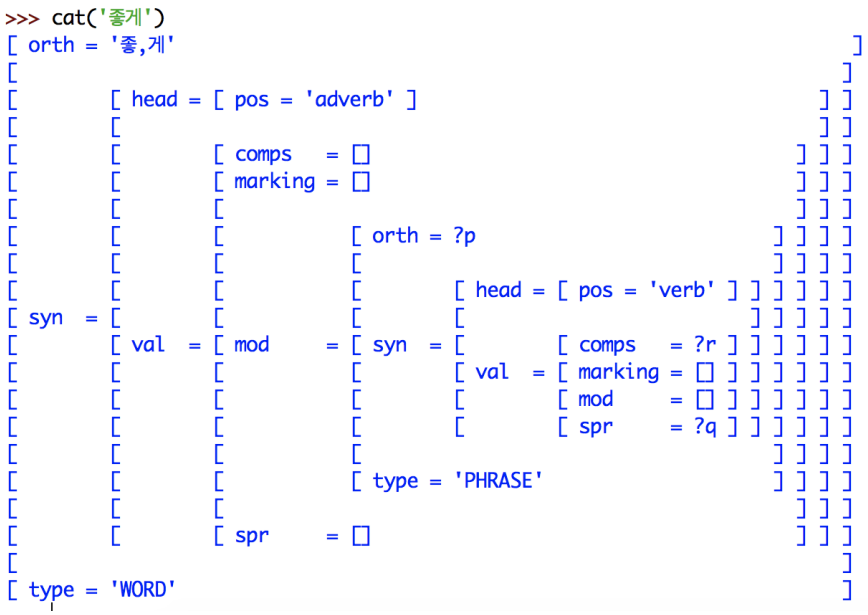


Figure 15: Constraint of Adverb 'well'

2.3 Phrase Rules

After lexical rules are applied and all lexemes are coverted into words or phrases, which are elements of syntax, the phrase rules are applied. Three kinds of phrase rules are implemented in this project: i) Head-Specifier rule, ii) Head-Complement rule and iii) Modifier rule.

2.3.1 Head-Specifier rule

Head-Speicifier rule allows one lexical entry can be combined with its specifier when its complement is satisfied.

Execution example

a. 강아지-는 잤-다(자-었-다) 'A dog slept'

dog-NOM sleep-Past-Declaration

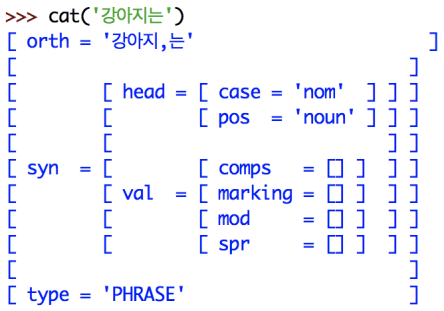


Figure 16: Constraint of nominative noun phrase

Please see Figure 10. for the verb phrase (잤다 *'slept'*).

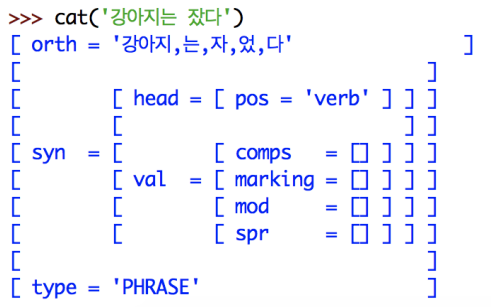


Figure 17: Constraint of 'a dog slept'

2.3.2 Head-Complement rule

Head-Complement rule allows one lexical entry can be combined with its complement.

Execution example

a. 강아지-를 잡-았-다 'caught a dog'

dog-ACC catch-Past-Declaration

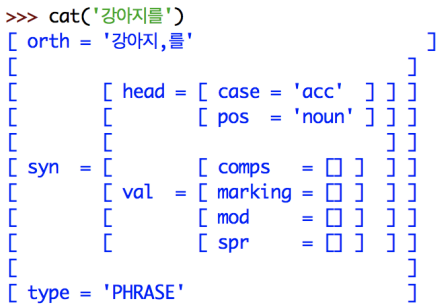


Figure 18: Constraint of accusative noun phrase

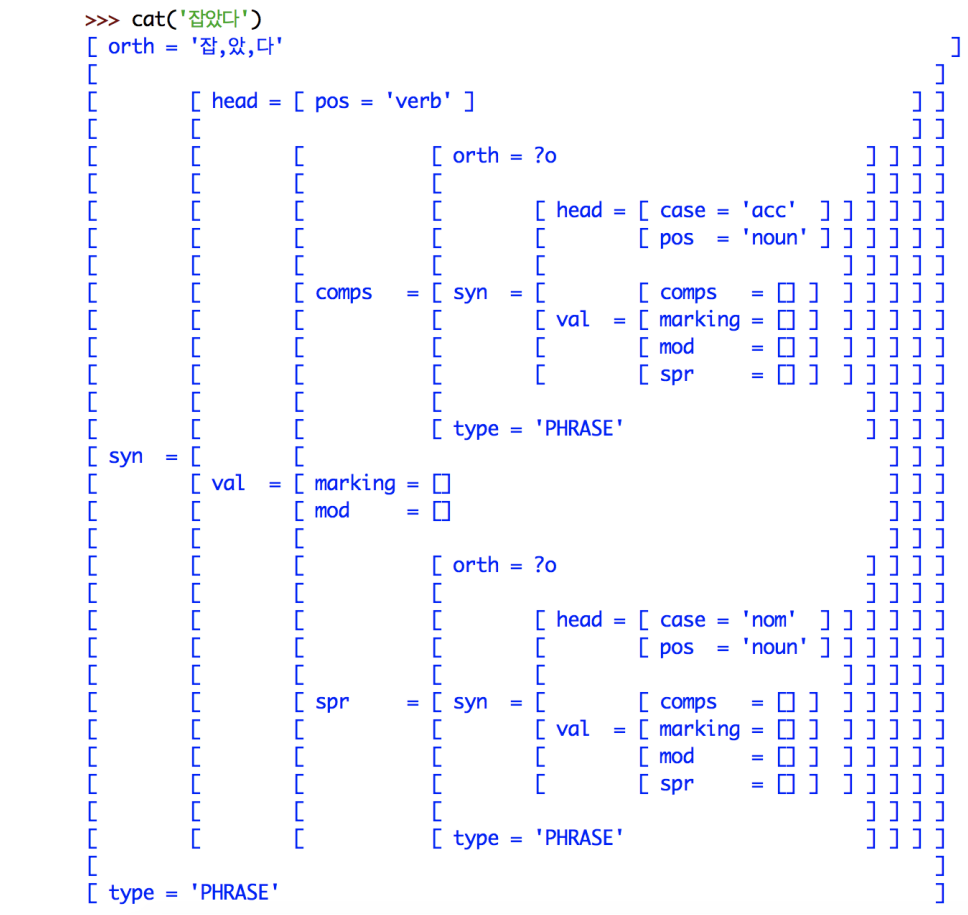
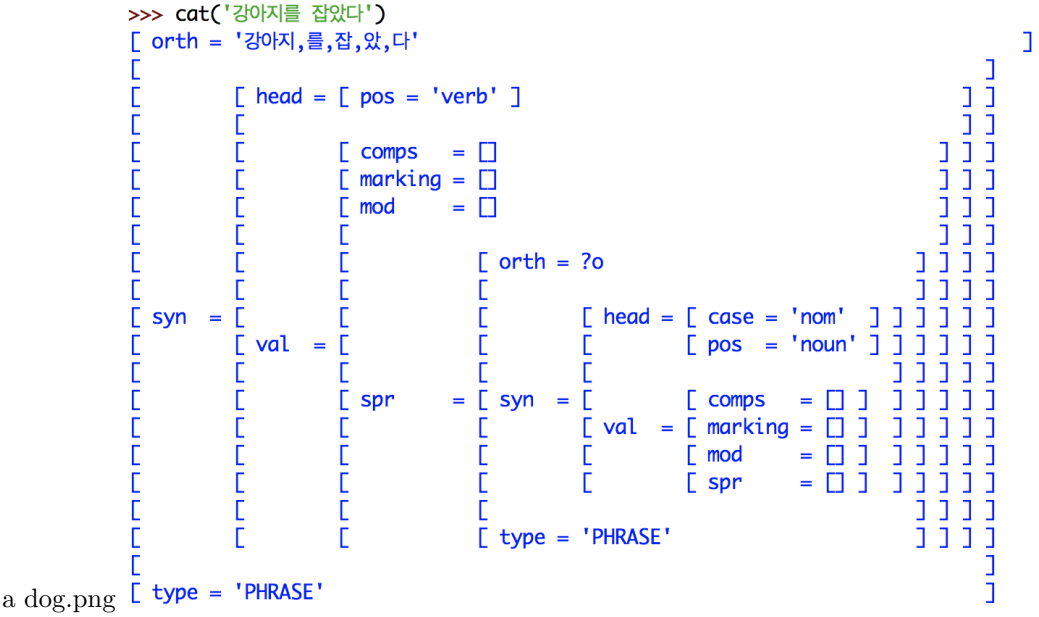


Figure 19: Constraint of 'caught'



a dog.png

Figure 20: Constraint of 'caught a dog'

2.3.3 Modifier rule

Modifier rule allows one lexical entry which have unempty 'mod' feature to be combined with the lexical entry that it can modify.

Execution example

a. 좋-게 잤-다(자-었-다) 'well slept'

nice-Adv sleep-Past-Declaration

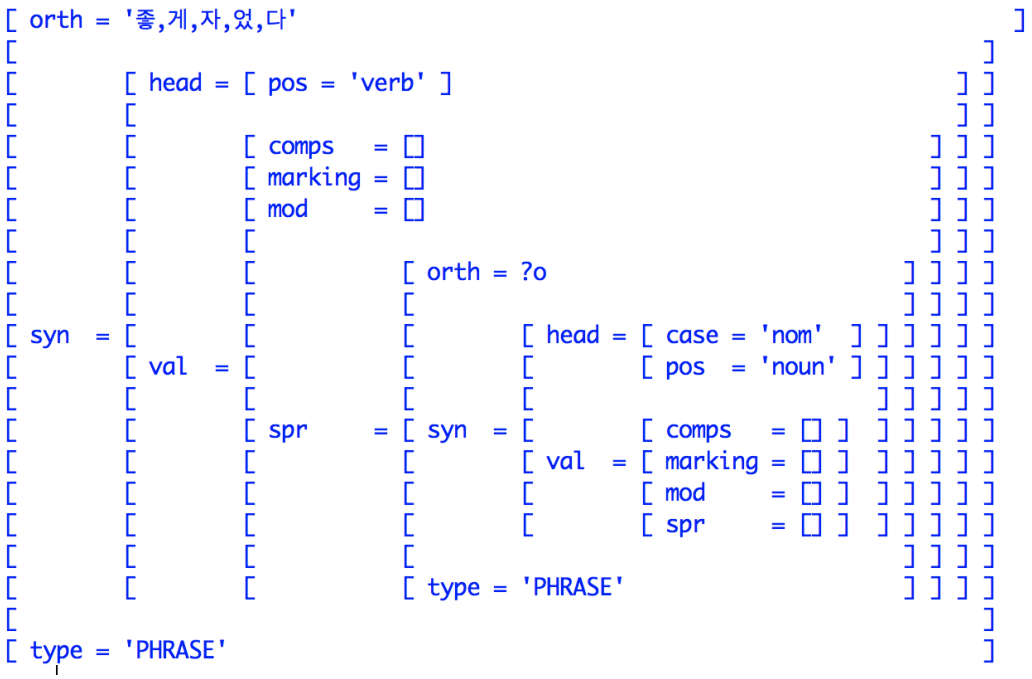


Figure 21: Constraint of 'slept well'

Please see Figure 15. for the adverb (좋게 *'well'*) and Figure 7. for the verb phrase (잤다 *'slept'*).

b. 좋-은 강아지 ‘a nice dog’

nice-Adj dog

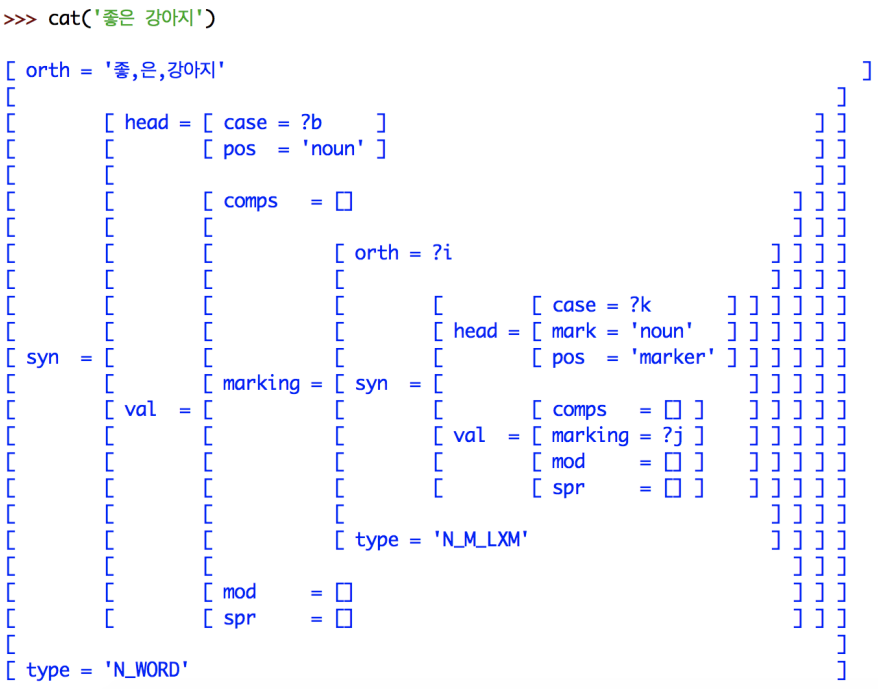


Figure 22: Constraint of ‘a nice dog’

Please see Figure 13. for the verb phrase (좋은 *‘good-Adj’*) and Figure5. for the noun (강아지 *‘dog’*).

3 More Execution examples

a. 크-ㄴ 고양이-는 작-은 쥐-를 빨리 잡-는-다 'big cat catches a mouse well'

big-Adj cat-NOM small-Adj mouse-ACC fast catch-Pres-Declaration

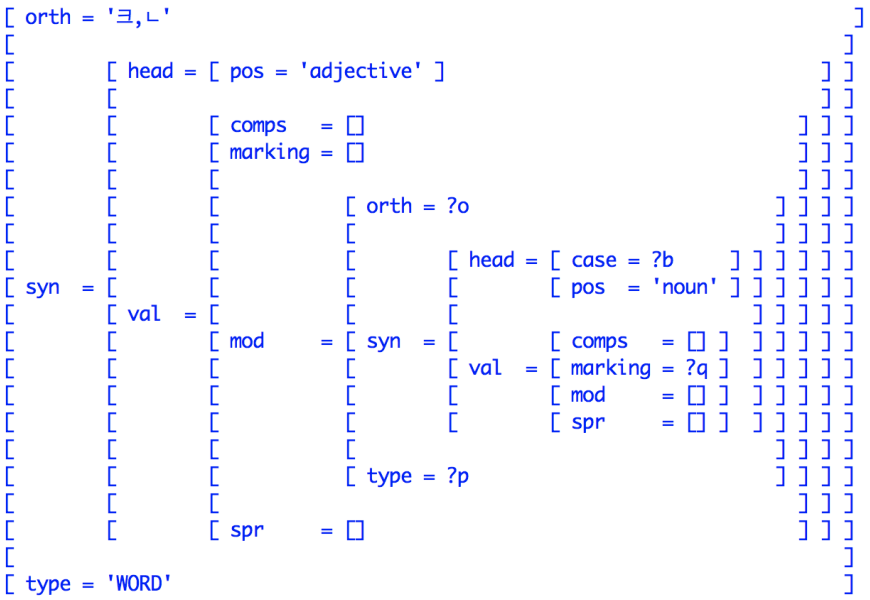


Figure 23: Constraint of 'big'

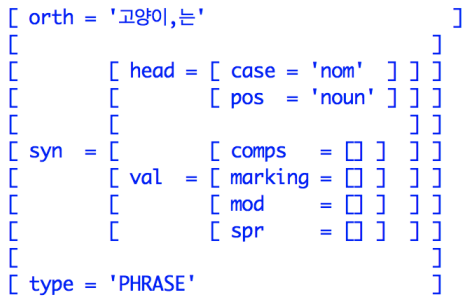


Figure 24: Constraint of 'cat-nom'

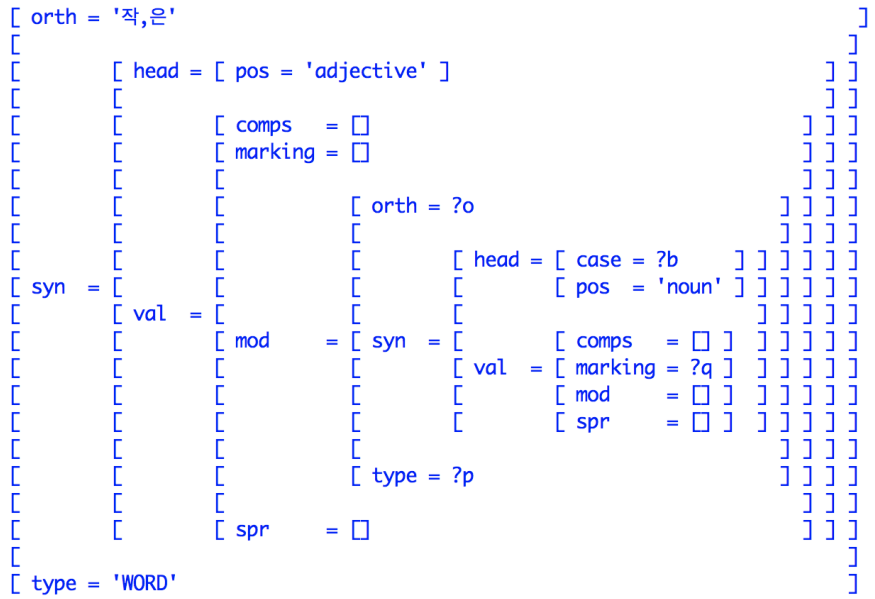


Figure 25: Constraint of 'small'

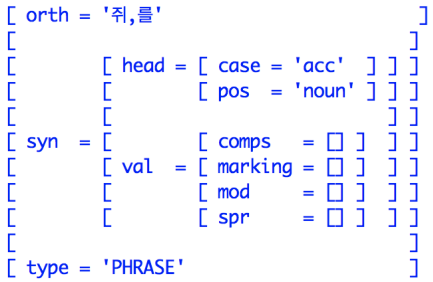


Figure 26: Constraint of 'mouse-acc'

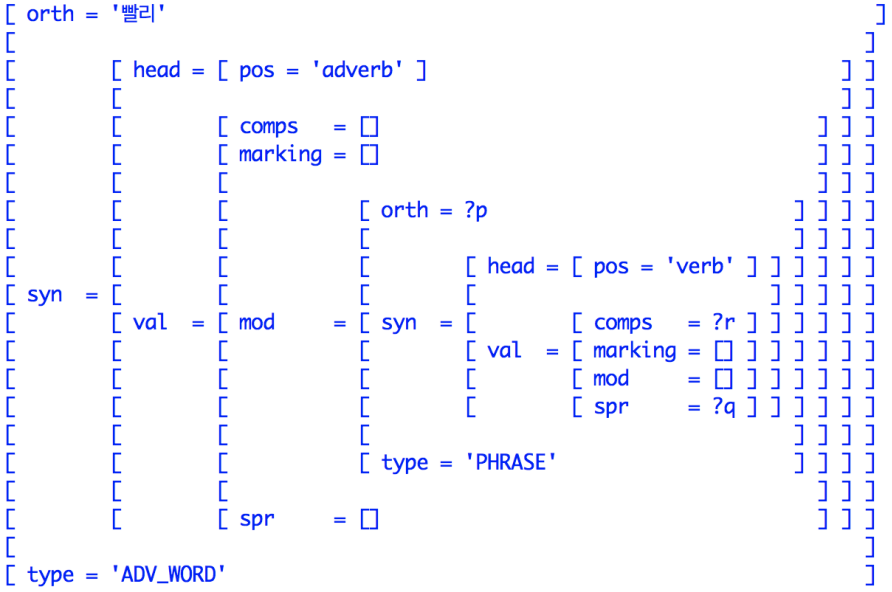


Figure 27: Constraint of 'fast'

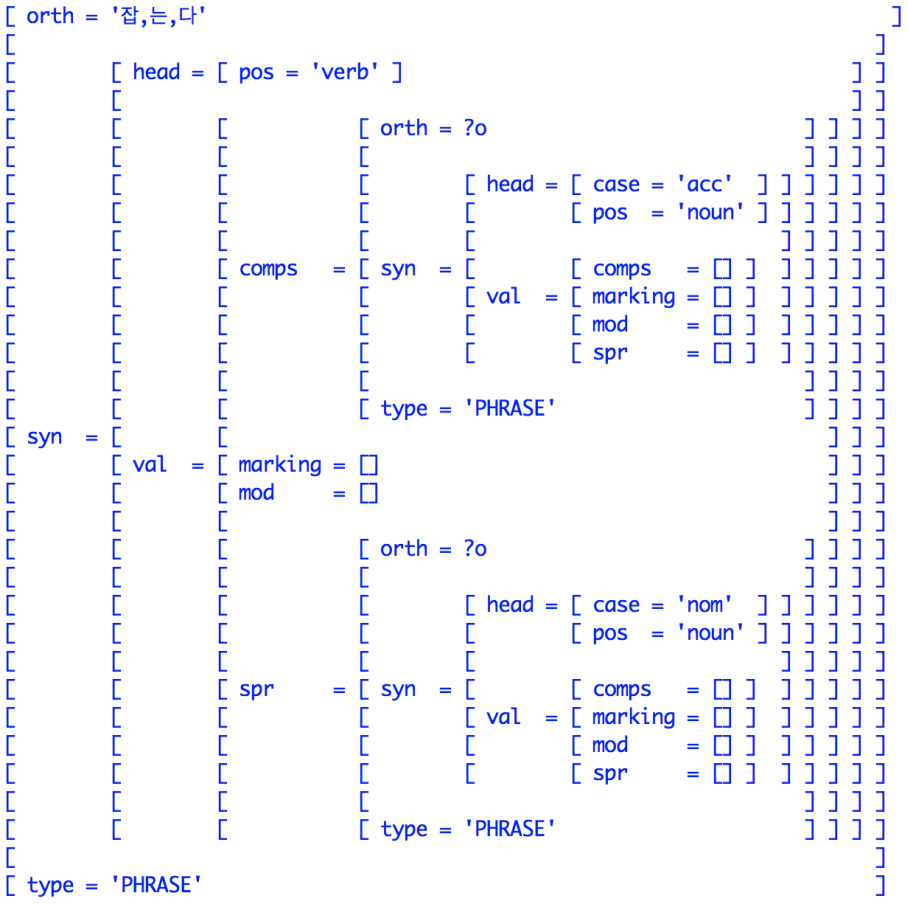


Figure 28: Constraint of 'catches'

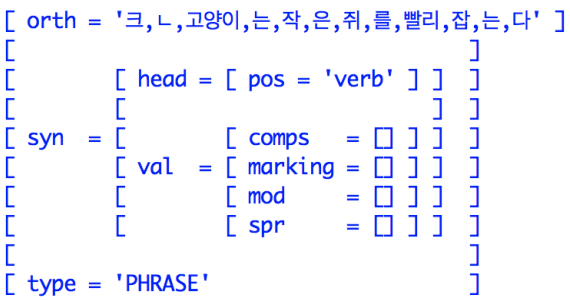


Figure 29: Constraint of the entire sentence

References

Kim, J.-B., & Yang, J. (2003). Korean phrase structure grammar and its implementations into the lkb system. In *Proceedings of the 17th pacific asia conference on language, information and computation* (pp. 88–97).

Park, E. L., & Cho, S. (2014). Konlpy: Korean natural language processing in python. In *Proceedings of the 26th annual conference on human & cognitive language technology* (pp. 133–136).