A Simple Syntax for Complex Semantics

Kiyong Lee
Department of Linguistics, Korea University
Seoul, 136-701 Korea
klee@korea.ac.kr

Abstract

As part of a long-ranged project that aims at establishing database-theoretic semantics as a model of computational semantics, this presentation focuses on the development of a syntactic component for processeing strings of words or sentences to construct semantic data structures. For design and modeling purposes, the present treatment will be restricted to the analysis of some problematic constructions of Korean involving semi-free word order, conjunction and temporal anchoring, and adnominal modification and antecedent binding.

The present work heavily relies on Hausser's (1999, 2000) SLIM theory for language that is based on surface compositionality, time-linearity and two other conditions on natural language processing. Time-linear syntax for natural language has been shown to be conceptually simple and computationally efficient. The associated semantics is complex, however, because it must deal with situated language involving interactive multi-agents. Nevertheless, by processing input word strings in a time-linear mode, the syntax can incrementally construct the necessary semantic structures for relevant queries and valid inferences.

The fragment of Korean syntax will be implemented in Malaga, a C-type implementation language that was enriched for both programming and debugging purposes and that was particluarly made suitable for implementing in Left-Associative Grammar. This presentation will show how the system of syntactic rules with constraining subrules processes Korean sentences in a step-by-step time-linear manner to incrementally construct semantic data structures that mainly specify relations with their argument, temporal, and binding structures.

1 Introduction

The main purpose of this work is to lay a syntactic basis for computational semantics. Despite the complexity of computational semantics, it is assumed that its underlying syntax must be simple. A simple syntax is advocated for complex semantics.

The task of computational semantics cannot be but complex because its ultimate aim is to model how human-machine communications are carried out by means of natural language. That of syntax can, however, be made simple at both the structural and the procedural level. The proposed syntactic module KoSyn, for instance, is implemented to be structurally simple, only consisting of a set of concatenation rules with some constraining subrules like valency check or parameter binding. It is also made procedurally simple because it processes linguistic input in a time-linear manner and builds up semantic data structures incrementally.

For this reason, the present talk will aim at showing how to minimalize the task of syntax by constructing the Korean syntactic module called KoSyn for testing and demonstration purposes. It is designed to generate a network of semantic data structures without deriving intermediate syntactic trees and also to represent them in a format that database-theoretic semantics can recognize and manipulate for the purpose of interpretation, production, and inference. This presentation will, however, be restricted to the discussion of representational issues for semantics.

By analyzing a fragment of Korean, this presentation will show in detail how KoSyn generates semantic structures for simple and conjoined sentences and also for adnominal clauses. It will also show how valency checking works for building argument structures, how nominal indexes are distributed over conjoined sentences, how events are temporally anchored, and how parameters are bound in adnominal constructions. All these processes are shown to be carried out linearly without backtracking.

2 Framework

This presentation assumes, as was claimed by Hausser (1999, 2001) and Lee (1999, 2000), that neither a naive set-theory or a Montagovian-type of logical frameworks can serve as an adequate metatheoretic basis for modeling

the use of natural language.[1] Instead, they both agree that some system of database construction and management theory must be adopted for modeling the computational procedures of human-machine communications that are carried out in real complex situations, thereby exchanging complex information through contextually tokenized language.[2]

The syntactic module KoSyn is a part of the interface module of database-theoretic semantics that constructs semantic data structures by processing linguistic signals and their information. Before storing semantic and other relevant data structures in a database, the interface module called LIPS takes in linguistic signals and their associated information in various representational forms, but converts them into a unique format that a database management system DMBS can recognize and manipulate. On the conceptual level, for instance, the relational model RDBMS can handle data structures represented in a table form only. One of the tasks of the interface engine LIPS is thus to provide an efficient procedure of representing linguistic information in a suitably required format. One possible candiate for such a format is a conventionally accepted feature structure format.[3]

2.1 Semantic Representation

Each atomic or basic sentence that consists of a verbal head and its complements is normally interpreted as expressing a proposition or carrying some prositional content. This content may then be partially represented in a feature structure of the following form:

[1]Specifically, Hausser (2001) argues that neither a truth-conditional or a use-conditional metalanguage-based semantics is suitable for constructing a computational model of natural language communication. Lee (2000) also claims that a theoretically well-founded database management system can be used as a model for constructing a communication engine that controls and manages information flows by processing language.

[2]Refer to Lee (1998) for the discussion of language situated in complex situations.

[3]Ordinary RDBMS cannot recognize feature structures, but object-oriented models can, since these models allow the embedding of tables into a larger table as attribute values.

(1) Semantic Structure in a FS Form

```
[PHON: ''string of words'',
 CAT: sentence,
 SEM_struc: <[REL: <>,
              ARG: <>,
              SIT: [],
              PRN: nil]>,
 COMP_stack: <>,
 PAR_binding: <>,
 NP_list: <>,
 C_list: <>]
```

The feature structure as represented in (1) provides each string of words PHON with information about its syntactic CATegory, SEMantic structure, COMPlement stack, PARameter binding, a processed NP list, and a processed Constituent list.

SEM_struc is then treated as taking a list of objects called *proplets* as value, where each proplet minimally consists of REL, ARG, SIT, and PRN and represents the content of an elementary proposition.[4] REL takes as value a list of pieces of information provided by a predicate, ARG a list of its arguments, and SIT a list of pieces of situational and temporal information.

The following is an example of a semantic data structure in a feature structure form:

(2) Semantic Structure Illustrated

```
[PHON: ''Mia loves me'',
 CAT: sentence,
 SEM_struc: <[REL: <act, ''love''>,
              ARG: <[np_index:1,
                     par_index: 1,
                     Role: agent,
```

[4]The term *proplet* was first coined by Hausser (1999) to represent the content of a (partially) saturated functor-argument structure or an elementary propositoinal concept token in a semantic database. Refer to Hausser (2001, 33ff, footnote 6) for the more accurate use of the term.

```
                    GF: subj],
                   [np_index:2,
                    par_index: 2,
                    Role: theme,
                    GF: obj]>,
              SIT: [T_index: <0>,
                    T_anchor: present],
              PRN: 1]>,
COMP_stack: <>,
PAR_binding: <>,
NP_list: <[np_index: 1,
           PHON: ''Mia'',
           CAT: noun,
           AGR: [Person: third,
                 Number: singular],
           SEM: [REF: <named, ''Mia''>,
                 Typ_of_OBJ: human]],
          [np_index: 2,
           PHON: ''me'',
           CAT: noun,
           AGR: [Person: first,
                 Number: singular,
                 Case: acc],
           SEM: [REF: <speaker>,
                 Typ_of_OBJ: human]]>]
```

Here, the REL:<act, ''love''> takes two arguments, par_index:1 and par_index:2 which are respectively linked to np_index:1 referring to Mia and np_index:2 referring to the speaker.This relation of love is also marked by SIT values as being temporally anchored to the present situation.

The list of COMP_stack in (2) is empty because each of its elements has been used up to saturate ARG by linking each par_index in ARG to an NP_list. The list of PAR_binding is also empty because there is no member of ARG with its par_index not anchored to some np_index.

2.2 Two Types of Lexicon

Syntax consists of two components: Lexicon and Rules. In Hausser's (1989) Left-Associative Grammar, there are also two types of lexicon: one is a basic lexicon and the other a derived expanded lexicon. The basic lexicon consists of morpheme-like items with minimally specified necessary lexical information. The derived expanded lexicon is generated by allomorphy rules from the basic lexicon and contains both allomorphic variants and derivational word forms with more enriched information.[5]

For example, the verb **love** can be listed at the basic lexicon, as in the following format:

(3) Basic Lexicon

```
[PHON: ''love'',
 CAT: verb,
 VAL: <<subj,agent>,<obj,theme>>,
 SEM: [REL:<act, ''love''>]]
```

From this entry in the basic lexicon, we can generate the expanded lexicon that contains the following derived lexical entry.

(4) Expanded Lexicon

```
[PHON: ''loves'',
 CAT: verb,
 Requires: [CAT: noun,
            AGR_feat:[Person: third,
                      Number: singular]]
 SEM: [REL: <act, ''love''>,
       ARG: <[par_index: 1,
              Role: agent,
              GF: subj],
             [par_index: 2,
              Role: theme,
              GF: obj]>,
       SIT: [T_anchor: present]]]
```

[5]For detailed discussion, refer to Lee (1999).

This double-deck lexical system can treat not only inflectional morphology, but also derivational morphology. For example, the Korean morphological system KoMor with its allomorphy rules can derive a verb 사랑하다 **sa.rang.ha.ta** "love" from the basic noun entry 사랑 **sa.rang** "love" and the verbal suffix 하다 **ha.ta** "do". Here are two lexical entries: one is basic and the other derived.

(5) Basic Entry

```
[Phon: "사랑",
 Cat: noun,
 Deriv: ha_acc,
 Sem: [Content: <act,"사랑_love">]]
```

(6) Derived Verb

```
malaga> ma 사랑하다
Analyses of
"사랑하다":
1: [Phon: "사랑하다",
    Segmentation: "사랑/하/다",
    Cat: verb,
    SEM: [REL: <사랑_love">,
          ARG: <[Par: 1,
                 Role: subj,
                 Case: nom],
                [Par: 2,
                 Role: theme,
                 Case: acc]>],
    Baseform: "사랑",
    VForm: terminal]
```

The derivational feature `Deriv: ha_acc` in the basic entry of the noun 사랑 **sa.rang** triggers the detailed specification of SEM features in the derived form.

2.3 Linear Processing

In Left-Associative Grammar, both morphological and syntactic rules are formulated in the same manner in `Malaga`, an acronym for a C-type implementation language for programming and debuggins purposes.[6] Basic conditions are placed on the grammar by the principles of surface compositionality and time-linearity and two other principles laid down by Hausser's (1999) SLIM theory of language. These principles do not permit any transformations or backtracking. They only allow (left-associative) linear processing and incremental composition.[7]

The set of grammatical rules must also match the set of program statements transparently at the level of implementation as well as at the level of conceptual design. Hence, the implemented grammar must follow the SLIM theory, processing strings of words or sentences linearly, while constructing the `SEM_struc` of a proplet or a proposition incrementally.

The notion of constituent structure has any significance in linear processing. The sentence **Mia loves me**, for instance, is not analyzed as having any conventional phrase structure tree. Nor is it analyzed as having a VP constituent. Instead, it is simply analyzed as a linearly concatenated string *((Mia loves) me)* of three words. The sentence **Mia really loves me** also undergoes the same linear processing. The topicalized sentence **Me, Mia really loves.** again is analyzed in the same manner.

Linear processing is achieved by a set of rules and another set of subrules. The main function of main rules is traffic control: these rules analyze input strings and put them in an appropriate stack according their grammatical functions. Nouns are mostly placed in `COMP_stack` and adverbs in `ADJUNCT_stack`, while verbs tokenize `SEM_struc` by specifying each of its underspecified or empty-valued features, `REL`, `ARG`, and `SIT`.

The subrules, on the other hand, function as constraints. One of the subrules in English syntax may check the subject-verb agreement when the noun **Mia** combines with the verb **loves** to form the string **Mia loves**.

[6] For the detailed description of the language and its use, refer to Beutel (2000).

[7] Refer to Nerbonne for the incremental processing in computational semantics. Note, however, that he does not favor the left-associative approach by stating: "Since the grammar is left-associative, the processing can also be incremental. This solution is formally sound, but its linguistic assumptions are heterodox. . . ."(477)

Another subrule can check the case agreement when the string **Mia loves** is to combine with the object noun **me** to produce the well-formed sentence **Mia loves me**, while blocking the ill-formed string **Mia loves I**.

In the process, various non-tokenized gaps in the `SEM_struc` are filled in. When the subject noun **Mia** combines with the verb **loves**, its first argument is tokenized with the index of the noun **Mia**. The second argument is tokenized when the object noun **me** is introduced. This process of tokenization is controlled by the subrule of valency check.

3 Analysis of Korean for Illustration

Since Korean is an agglutinative language, its nominal particles and verbal endings carry important syntactic and semantic information. The case particles, for instance, play an important role in forming a sentence by conjoining a list of nominal complements with a verbal head, and the tense endings in anchoring the temporal frame of a proposition conveyed by a sentence. Focusing on the roles of case particles, the present presentation will analyze a small fragment of Korean involving [1] word order and argument structure, [2] conjunction and parameter anchoring, and [3] adnominal construction and parameter binding.[8]

Before starting the main analyses of Korean, the main mechanism of `KoSyn` will be briefly illustrated with the copular construction, the agglutinative property of Korean is well demonstrated by the copular construction. In Korean, the copula **.i** is not a verb by itself, but becomes a predicate when it is combined with a noun and also with a terminal verbal ending. The noun 미인 **mi.in** "beauty, beautiful woman" may combine with the copular suffix 이 **.i** and then with the terminal ending 다 **ta** to form an adjectival verb 미인이다 **mi.in.i.ta** "be a beauty".

This analysis can be treated at the morphological level, as shown in the following output by `ma`, morphological analysis through `malaga`:

[8]For anlayzing these constructions, I heavily rely on Chang (1993, 1996) as the main reference sources.

(7) Derivation of a Predicative Noun

```
malaga> ma 사랑아다
Analyses of "미인이다":
1: [Phon: "미인이다",
    Segmentation: "미인/이/다",
    Cat: adj,
    SEM: [REL: <characterized_as, "a beauty">,
          ARG: <[Par: 1,
                 Role: subj,
                 Case: nom]>,
          Type_of_Obj: human,
          Tense: present],
    Baseform: "미인",
    STyp: declarative,
    VForm: terminal,
    SLv: plain,
    Mode: bse]
```

The concatenated string of 미인/이/다 is treated as having the SEMantic structure with its REL and one argument with its Role: subj and Case: nom. Now, this string called predicative noun can take a nominative-marked noun as its subject, yielding a sentence like the following.

(8) Demo 0

```
malaga> sa 미아가 미인이다
Analyses of "미아가 미인이다":
1: [Phon: "미아가 미인이다",
    Cat: s,
    SEM_struc: <[prop_index: 1,
                 REL: <human, characterized_as, "a beauty">,
                 ARG: <[np_index: 1,
                        Par: 1,
                        Role: subj,
                        Case: nom]>,
                 SIT: [T_index: <0>,
```

```
                    T_anchor: present,
                    T_state: nil,
                    SLv: nil]]>,
    comp_stack: <>,
    vform_stack: <terminal>,
    par_binding: <>,
    np_list: <[np_index: 1,
              Phon: "미아가",
              Cat: noun,
              SEM: [REF: <named, "미아_Mia">,
                    Type_of_Obj: human],
              Case: nom]>]
```

This analysis is obtained through the following procedure.

First, the noun 미아가 **mi.a.ka** is introduced by the NP_introduction rule with the following analysis:

(9) NP introduction

```
malaga> sa 미아가
Analyses of "미아가":
2: [Phon: "미아가",
    SEM_struc: <>,
    comp_stack: <[np_index: 1,Case: nom]>,
    vform_stack: <>,
    par_binding: <>,
    np_list: <[np_index: 1,
              Phon: "미아가",
              Cat: noun,
              SEM: [REF: <named, "미아_Mia">,
                    Type_of_Obj: human],
              Case: nom]>,
    constituent_list: <[Phon: "미아가",
                        Cat: noun,
                        SEM: [REF: <named, "미아_Mia">,
                              Type_of_Obj: human],
                        Case: nom]>]
```

Here, comp_stack is filled in with the feature structure [np_index: 1, Case: nom] provided by the introduction of the nominative-case marked noun 미아가 **mi.a.ka**.

Secondly, this noun combines with a verb which is being introduced by the head rule. Then, the subrule valency_check checks if the feature structure in comp_stack matches any of the arguments listed in ARG. Since there is one argument whose Case value matches the feature Case: nom in comp_stack, its Par: 1 is anchored to np_index: 1, thus referring to Mia.

This procedure illustrates that, in KoSyn, the acceptable SEM_struc of the copular construction 미아가 미인이다 "Mia is a beauty" is obtained without constructing any intermediate constituent structure.

3.1 Word Order and Argument Structure

Korean is known to be a head-final language, thus allowing free word order except that the head must occur at the end.[9] The following sentences illustrate this relative free ordering.

(10) Free Word Order Illustrated

a. 미아가 사과를 먹었다
mia-ka sa.kwa.lul mek.ess.ta
mia-NOM apple-ACC eat-PAST
''Mia ate an apple''

b. 사과를 미아가 먹었다
apple-ACC mia-NOM eat-PAST

Despite their difference in word order, these two sentences carry the same propositional content, as analyzed by KoSyn.

[9]If the head is of a terminal form, then its complements or adjuncts may occur after the head. For example, 예쁘다, 미아가. **yey.ppu.ta mia-ka** "is-pretty, Mia".

(11) Demo 1

```
Analyses of "미아가 사과를 먹었다":
1: [Phon: "미아가 사과를 먹었다",
    Cat: s,
    SEM_struc: <[prop_index: 1,
                 REL: <action, "먹다_eat">,
                 ARG: <[np_index: 1,
                        Par: 1,
                        Role: agent,
                        Case: nom],
                       [np_index: 2,
                        Par: 2,
                        Role: theme,
                        Case: acc]>,
                 SIT: [T_index: <0>,
                       T_anchor: past,
                       T_state: nil,
                       SLv: plain]]>,
    comp_stack: <>,
    vform_stack: <terminal>,
    par_binding: <>,
    np_list: <[np_index: 1,
               Phon: "미아가",
               Cat: noun,
               SEM: [REF: <named, "미아_Mia">,
                     Type_of_Obj: human],
               Case: nom],
              [np_index: 2,
               Phon: "사과를",
               Cat: noun,
               SEM: [REF: <characterized_as, "사과", "apple">,
                     Type_of_Obj: fruit],
               Case: acc]>]
```

In the above analysis, the first argument which is tagged with Par: 1 and plays the agent role is anchored to np_index: 1, thereby referring

to Mia. The second argument, on the other hand, is tagged with Par: 2 and plays the theme role. It is then anchored to np_index: 1 and thus is linked to a kind of fruit characterized as apple.

(12) Demo 2

```
Analyses of "사과를 미아가 먹었다":
1: [Phon: "사과를 미아가 먹었다",
    Cat: s,
    SEM_struc: <[prop_index: 1,
                 REL: <action, "먹다_eat">,
                 ARG: <[np_index: 1,
                        Par: 2,
                        Role: theme,
                        Case: acc],
                       [np_index: 2,
                        Par: 1,
                        Role: agent,
                        Case: nom]>,
                 SIT: [T_index: <0>,
                       T_anchor: past,
                       T_state: nil,
                       SLv: plain]]>,
    comp_stack: <>,
    vform_stack: <terminal>,
    par_binding: <>,
    np_list: <[np_index: 1,
               Phon: "사과를", Cat: noun,
               SEM: [REF: <characterized_as, "사과", "apple">,
                     Type_of_Obj: fruit],
               Case: acc],
              [np_index: 2,
               Phon: "미아가", Cat: noun,
               SEM: [REF: <named, "미아_Mia">,
                     Type_of_Obj: human],
               Case: nom]>]
```

Here, the argument Par: 2 which plays the theme role is first anchored to np_index: 1, namely apple and the argument Par: 1 with its Role: agent is then anchoredd to np_index: 2, Mia. These two analyses show that the order of anchoring arguments has changed, but that their propositonal content has been preserved.

Despite the relative free word order in Korean, some of the so-called double subject or double object constructions do not allow the scrambling of two nouns each marked with a nominative case particle 이/가 **.i/ka** or with an accusative case particle 을/를 **.ul/lul**.[10]

The following two sentences, for instance, are not synonymous.

(13) Switching Disallowed

a. 물이 얼음이 되었다
   mwul.i .el.um.i toy.ess.ta
   water-NOM ice-NOM become-PAST
   ''water became ice''
b. 얼음이 물이 되었다
   ''ice became water''

It is shown that KoSyn analyzes these sentences correctly.

(14) Double Subject Construction: Analysis 1

```
Analyses of "물이 얼음이 되었다":
1: [Phon: "물이 얼음이 되었다",
    Cat: s,
    SEM_struc: <[prop_index: 1,
                 REL: <"되다_become">,
                 ARG: <[np_index: 1,
                        Par: 1,
                        Role: subj,
                        Case: nom],
                       [np_index: 2,
                        Par: 2,
```

[10]For the discussion of these phenomena, refer to Chang (1993, 1996).

```
                        Role: theme,
                        Case: nom]>,
                  SIT: [T_index: <0>,
                        T_anchor: past,
                        T_state: nil,
                        SLv: plain]]>,
    comp_stack: <>,
    vform_stack: <terminal>,
    par_binding: <>,
    np_list: <[np_index: 1,
               Phon: "물이",
               Cat: noun,
               SEM: [Content: <"물", "water">]
               Case: nom],
              [np_index: 2,
               Phon: "얼음이",
               Cat: noun,
               SEM: [Content: <"얼음", "ice">]
               Case: nom]>]
```

(15) Double Subject Construction: Analysis 2

```
Analyses of "얼음이 물이 되었다":
1: [Phon: "얼음이 물이 되었다",
    Cat: s,
    SEM_struc: <[prop_index: 1,
                 REL: <"되다_become">,
                 ARG: <[np_index: 1,
                        Par: 1,
                        Role: subj,
                        Case: nom],
                       [np_index: 2,
                        Par: 2,
                        Role: theme,
                        Case: nom]>,
                 SIT: [T_index: <0>,
                       T_anchor: past,
```

```
                    T_state: nil,
                    SLv: plain]]>,
comp_stack: <>,
vform_stack: <terminal>,
par_binding: <>,
np_list: <[np_index: 1,
           Phon: "얼음 이",
           Cat: noun,
           SEM: [Content: <"얼음", "ice">],
           Case: nom],
          [np_index: 2,
           Phon: "물 이",
           Cat: noun,
           SEM: [Content: <"물", "water">],
           Case: nom]>]
```

These analyses show that each sentence is correctly analyzed. In the first analysis, the argument with `Role: subj` is anchored to water and the argument with `Role: theme` to ice, thus meaning that water became ice. In the second analysis, other hand, the argument with `Role: subj` is anchored to ice and then the argument with `Role: theme` to water, now meaning that ice became water. Here, the switching of two nominative-marked nous in these sentences has resulted in a difference in meaning.

3.2 Conjunction: Referential and Temporal Anchoring

The following simple conjoined sentence illustrates two issues: one involves referential anchoring and the other temporal anchoring.

(16) Conjoined Sentence

```
미아가 사과를 먹고 잤다
mi.a.ka sa.kwa.lul mek.ko cass.ta
Mia-Nom apple-ACC eat-CONJ sleep-PAST-DECL
''Mia ate an apple and slept''
```

First, the subject of 잤다 **cass.ta** "slept" is understood to be Mia, the subject of the preceding conjunct. KoSyn anchors the subject parameter

of the verb 잤다 **cass.ta** to Mia. Secondly, the verbal head 먹고 **mek.ko** "eat-CONJ" of the first conjunct is not tense-marked. But it will be shown in KoSyn that its temporal index is anchored to the past. Here is a brief analysis of the sentence.

(17) Anchoring

```
1: [Phon: "미아가 사과를 먹고 잤다",
    Cat: s,
    SEM_struc: <[prop_index: 1,
                 REL: <action, "먹다_eat">,
                 ARG: <[np_index: 1,
                        Par: 1,
                        Role: agent,
                        Case: nom],
                       [np_index: 2,
                        Par: 2,
                        Role: theme,
                        Case: acc]>,
                 SIT: [T_index: <0>,
                       T_anchor: nil,
                       T_state: nil,
                       SLv: nil]],
                [prop_index: 2,
                 REL: <state, "자다_sleep">,
                 ARG: <[np_index: 1,
                        Par: 1,
                        Role: agent,
                        Case: nom]>,
                 SIT: [T_index: <0, 1>,
                       T_anchor: past,
                       T_state: nil,
                       SLv: plain]]>,
    comp_stack: <>,
    vform_stack: <terminal>,
    par_binding: <>,
    np_list: <[np_index: 1,
```

```
 Phon: "미아가",
 Cat: noun,
 SEM: [REF: <named, "미아_Mia">,
       Type_of_Obj: human],
 Case: nom],
[np_index: 2,
 Phon: "사과를",
 Cat: noun,
 SEM: [REF: <characterized_as, "사과", "apple">,
       Type_of_Obj: fruit],
 Case: acc]>]
```

First, note that SEM_struc now contains two proplets: one is marked with prop_index: 1 and the other with prop_index: 2. In the first proplet, REL: <action, ''먹_eat''> holds between two arguments: one argument with Role: agent is anchored to np_index: 1, Mia, and another argument with Role: theme to np_index: 2, apple. In the second proplet, the sole argument with Par: 1 is anchored to np_index: 1, Mia, who is the subject of the first conjunct.

As for the temporal anchoring, the temporal index T_index: <0> in the first proplet is not anchored. But in the second proplet, the temporal index T_index: <0,1> is anchored to the past and the value of the temporal index of the first proplet is contained in the temporal index value of the second proplet. Hence, the temporal index of the first proplet is also anchored to the past.

This analysis shows that both referential and temporal anchorings can easily be handled in KoSyn. It should, however, be noted that temporal anchoring is achieved not by some syntactic process, but by inference on the feature structures of SIT in the two proplets.

3.3 Adnominal Modification and Parameter Binding

One type of adnominal modification in Korean corresponds to relativization in English. Here is an example:

(18) Relative Adnominal Modification

```
미아가 김을 사랑하는 교수를 사랑한다
mi.a.ka kim.ul sa.rang.ha.nun kyo.swu.lul sa.rang.han.ta
Mia-nom Kim-acc love  prof-acc love
''Mia loves a professor who loves Kim''
```

This examples raises two issues: one issue concerns the binding of the parametric Subject of the adnominal verb 사랑하는 and another concerns the problem of avoiding backtracking when the nominative-marked noun 미아가 is erroneously analyzed as the Subject of the adnominal verb 사랑하는 as is here.[11]

The following malaga analysis shows that these problems have been solved in KoSyn.

(19) Malaga Analysis

```
Analyses of "미아가 김을 사랑하는 교수를 사랑한다":
1: [Phon: "미아가 김을 사랑하는 교수를 사랑한다",
    Cat: s,
    SEM_struc: <[prop_index: 1,
                 SEM: <act, "사랑_love">,
                 ARG: <[np_index: 2,
                        Par: 2,
                        Role: theme,
                        Case: acc],
                       [Par: 1,
                        Role: subj,
                        Case: nom]>,
                 SIT: [T_index: <0>,
                       T_anchor: overlapping,
                       T_state: nil,
                       SLv: nil]],
                [prop_index: 2,
                 SEM: <act, "사랑_love">,
```

[11]Such an analysis must be accepted if the antecedent is a factive noun, as in 미아가 김을 사랑하는 이유 "reason that Mia loves Kim".

```
                ARG: <[np_index: 1,
                       Par: 1,
                       Role: subj,
                       Case: nom],
                      [np_index: 3,
                       Par: 2,
                       Role: theme,
                       Case: acc]>,
                SIT: [T_index: <0, 1>,
                      T_anchor: present,
                      T_state: nil,
                      SLv: nil]]>,
comp_stack: <>,
vform_stack: <terminal>,
par_binding: <[prop_index: 1,
               np_index: 3,
               Par: 1,
               Role: subj,
               Case: nom]>,
np_list: <[np_index: 1,
           Phon: "미아가",
           Cat: noun,
           SEM: [REF: <named, "미아_Mia">,
                 Type_of_Obj: human],
           Case: nom],
          [np_index: 2,
           Phon: "김을",
           Cat: noun,
           SEM: [Content: <named, "김_Kim">,
                 Type_of_Obj: human,
                 SFeat: KorName],
           Case: acc],
          [np_index: 3,
           Phon: "교수들",
           Cat: noun,
           SEM: [Content: <characterized_as, "교수", "professor">
```

```
                    Type_of_Obj: human],
               Case: acc]>]
```

In this analysis, SEM_struc has two proplets: the proplet with prop_index: 1 is generated by the adnominal modifier 김을 사랑하는 "that loves Kim" and the other proplet with proplet_index: 2 represents the content of the main clause 미아가 . . . 교수를 사랑한다 "Mia loves the professor . . .". In the first proplet, there is an unbound parameter marked with Par: 1 and Role: subj. This cannot be bound till the antencedent 교수를 **kyo.swu.lul** "professor" is introduced. Hence, it is stored in the list of par_binding, waiting to be bound. But here it is found bound to np_index: 3 and then is linked to some element in NP_list that refers to some professor.

In the second proplet, each of the two arguments is properly saturated. The first argument with Role: subj is linked to np_index: 1, Mia, and the second argument with Role: theme to np_index: 3, which refers to some professor.

The iterated adnominal construction as in the following example is also shown to be properly analyzed by KoSYn.second

(20) Exercise

```
malaga> sa 미아가 먹은 예쁜 사과를 용이 먹었다
Analyses of "미아가 먹은 예쁜 사과를 용이 먹었다":
1: [Phon: "미아가 먹은 예쁜 사과를 용이 먹었다",
    Cat: s,
    SEM_struc: <[prop_index: 1,
                 REL: <action, "먹다_eat">,
                 ARG: <[np_index: 1,
                        Par: 1,
                        Role: agent,
                        Case: nom],
                       [Par: 2,
                        Role: theme,
                        Case: acc]>,
                 SIT: [T_index: <0>,
```

```
                    T_anchor: prior,
                    T_state: nil,
                    SLv: nil]],
                [prop_index: 2,
                 REL: <"예쁘다_pretty">,
                 ARG: <[Par: 1,
                        Role: subj,
                        Case: nom]>,
                 SIT: [T_index: <0, 1>,
                    T_anchor: overlapping
                    T_state: nil,
                    SLv: nil]],
                [prop_index: 3,
                 REL: <action, "먹다_eat">,
                 ARG: <[np_index: 2,
                        Par: 2,
                        Role: theme,
                        Case: acc],
                       [np_index: 3,
                        Par: 1,
                        Role: agent,
                        Case: nom]>,
                 SIT: [T_index: <0, 1, 2>,
                    T_anchor: past,
                    T_state: nil,
                    SLv: plain]]>,
comp_stack: <>,
vform_stack: <terminal>,
par_binding: <[prop_index: 1,
               np_index: 2,
               Par: 2,
               Role: theme,
               Case: acc],
              [prop_index: 2,
               np_index: 2,
               Par: 1,
```

```
                Role: subj,
                Case: nom]>,
np_list: <[np_index: 1,
           Phon: "미아가",
           Cat: noun,
           SEM: [REF: <named, "미아_Mia">,
                 Type_of_Obj: human],
           Case: nom],
          [np_index: 2,
           Phon: "사과를",
           Cat: noun,
           SEM: [REF: <characterized_as, "사과", "apple">,
                 Type_of_Obj: fruit],
           Case: acc],
          [np_index: 3,
           Phon: "용이",
           Cat: noun,
           SEM: [REF: <named, "용_Yong">,
                 Type_of_Obj: human],
           Case: nom]>]
```

The checking of referential anchorings in the above analysis is left for an exercise. The temporal anchoring of the first adnominal clause, however, needs some attention. The `T_index: <0>` in the first proplet is anchored to be prior to other events. Furthermore, since other temporal indexes are anchored to the past as shown in the final proplet, the event of the first proplet is interpreted as preceding the events of other proplets.

4 Conclusion

As shown in this presentation, many of the problems in Korean syntax can be solved with a time-linear syntax that constructs semantic structures in an incremental manner. But there still remain more problems unsolved. Syntax still should remain simple without multiplicating rules and constraints. This optimism can be kept alive as long as one insists on the old belief that the job of syntax is to create complex trees.

I do not, however, believe that a theory is a belief. I don't have a belief in any theory, for there is no complete theory. If it were, it would be a dogma and no longer be a part of science. No grammar can be complete either. If it were, it would no longer be a part of linguistics and linguists must retire, as I will be planning to do.

5 Acknowledgements

For drafting this presentation, I am very much indebted to Roland Hausser, Suk-Jin Chang, and Jae-Woong Choe for their constructive and encouraging comments and to Jungha Hong for his invaluable help in implementing the Korean syntax *Kosyn5.6* based on Malaga5.6. I would like to thank all the executive members of the Korean Society for Language and Information, particularly Minhaeng Lee, Secretary of PACLIC16, and Ik-Hwan Lee, Chair of its Programming Committee, for inviting me to enjoy the conference as a keynote speaker. I also would like to thank my wife Ryun, who kept awake serving me tea while I was trying to finish this draft till five o'clock in the morning on Sunday at our country home.

References

[1] Beutel, Björn (2000), "Malaga Version 5.6: An Implementation Language for Left-Associative Grammar"(unpublished), Erlangen: Universität Erlangen-Nürnberg, Abteilung Computerlinguistik.

[2] Chang, Suk-Jin (1993), *Information-Based Korean Grammar* [written in Korean], Seoul: Language and Information and Hanshin.

[3] Chang, Suk-Jin (1996), *Korean*, Amsterdam: John Benjamins.

[4] Hausser, Roland (1989). *Computation of Language: An Essay on Syntax, Semantics and Pragmatics in Natural Man-Machine Communication*, Berlin: Springer-Verlag.

[5] Hausser, Roland (1999), *Foundations of Computational Linguistics: Man-Machine Communication in Natural Language*, Berlin: Springer.

[6] Hausser, Roland (2001), "Database semantics for natural language," *Artificial Intelligence* 130(2001), 27-74.

[7] Lee, Kiyong (1998), *Situation and Inforamtion: Situation Semantics*, Seoul: Taehaksa.

[8] Lee, Kiyong (1999), *Computational Morphology* [written in Korean], Seoul: Korea University Press.

[9] Nerbonne, John (1996), "Computational Semantics - Linguistics and Processing", in Shalom Lappin (ed.), *The Handbook of Contemporary Semantic Theory*, 461-484, Oxford: Blackwell.