Treating Quantifiers in Database Semantics

Jae-Woong CHOE
Korea University
Dept. of Linguistics
jchoe@korea.ac.kr

Roland HAUSSER
Universität Erlangen-Nürnberg
Abt. Computerlinguistik (CLUE)
rrh@linguistik.uni-erlangen.de

Abstract

This paper analyzes the syntactic and semantic structure of noun phrases in English and Korean, using the time-linear derivations of Database Semantics.[1] In comparison with Predicate Calculus, which handles the semantics of determiners like *some* and *all* at the highest level of the logical syntax, Database Semantics takes the alternative approach of specifying their semantics as atomic values in the feature structures representing noun phrases. This not only avoids well-known difficulties of the classical approach, such as unwanted scope ambiguities (Copestake et al. 2001) and problems binding certain variables (Geach 1969, Kamp & Reyle 1993), but also opens the way to concentrate on important linguistic aspects of complex noun phrases, namely agreement in English, and the alternative word orders and the rather free distribution of case markers inside the NP in Korean. Given that the internal structure of Japanese NPs is very similar to that of Korean, we believe that our analysis can be easily extended to include Japanese as well.

1 Interpretation of Quantifiers in Predicate Calculus

Since Russell's 1905 celebrated analysis of definite descriptions in terms of predicate calculus, the most widely used semantic interpretation of determiners like the, all, some, every, several, and the numerals one, two, three, etc., is by means of the familiar quantifiers $\exists$ and $\forall$. Consider the following example:

1.1 Predicate Calculus analysis of All girls sleep

$\forall x\ [girl(x) \rightarrow sleep(x)]$

The model-theoretic interpretation of such a formula is defined with respect to a model and a variable assignment. Following Montague 1974, the model is defined as a tuple (A,F), where A is a set of individuals, for example $\{a_0, a_1, a_2, a_3\}$, and F is an assignment function which assigns to every constant in the formal language an element of 2^A (i.e. the power set of A) as an interpretation. For example F(girl) might be defined in @ as $\{a_1, a_2\}$ and F(sleep) as $\{a_0, a_2\}$. This means intuitively that the individuals a_1 and a_2 in the model @ are girls, while the individuals a_0 and a_2 are sleeping.

The dependence of the truth-value of the formula on the actual definition of the model is represented by Montague as follows:

[1] As the name of a specifi c scientifi c theory, the term Database Semantics is written with initial capital letters. This use is distinct from referrring to generic issues, for example semantic constraints on databases (cf. Bertossi, Katona, Schewe, & Thalheim (eds.) 2003).

1.2 INTERPRETATION RELATIVE TO A MODEL

$\forall x\ [girl(x) \rightarrow sleep(x)]^{@,g}$

The quantifier $\forall$ is interpreted by means of the variable assignment g: the whole formula is true relative to the model @, if it holds *for all possible* variable assignments g' that the formula without the outermost quantifier is true.

1.3 ELIMINATION OF THE QUANTIFIER

$[girl(x) \rightarrow sleep(x)]^{@,g'}$

The purpose of eliminating the quantifier is to reduce the formula to propositional calculus and its truth-tables (cf. Bochenski 1961). This is possible by systematically assigning all possible values in the set of individuals A to the variable x and determining the truth-value of the subformulas girl(x) and sleep(x) for each assignment. Thus, g' first assigns to x the value a_0, then the value a_1, etc. Given the definition of the model, we can now check for each such assigment whether or not it makes the formula 1.3 true.

For example, the first assignment g'(x) = a_0 makes the formula true: a_0 is not in the set denoted by F(girl) in @; therefore, based on the truth-table of $p \rightarrow q$ in propositional calculus, the formula in 1.3 is true for this assignment. The second assignment g'(x) = a_1, in contrast, makes the formula in 1.3 false: a_1 is in the set denoted by F(girl), but not in F(sleep) in @. Having shown that *not all* variable assignments g' make the formula in 1.3 true, the interpretation of the formula in 1.1 is determined to be false relative to @. Given how the model @ = (A,F) was defined, this is in accordance with intuition.

2 Interpretation of Determiners in Database Semantics

The linguistic analysis of determiners requires not only a semantic interpretation, but also a handling of (noun phrase) internal and external agreement. In English, internal agreement is illustrated by the fact that every girl is grammatical while *every girls is not. External agreement is illustrated by the fact that every girl sleeps is grammatical, while *every girl sleep is not.

In preparation of our comparison of Predicate Calculus (PC) and Database Semantics (DBS) regarding the treatment of determiners in Section 3, let us consider the sentence every man loves a woman. In DBS, the first step of a grammatical analysis is the lexical analysis of the word forms:

2.1 LEXICAL ANALYSIS OF every, man, loves, a, AND woman

sur: every	sur: man	sur: loves	sur: a	sur: woman
noun: n_1	noun: *man*	verb: *love*	noun: n_1	noun: *woman*
cat: sn′ snp	cat: sn	cat: ns3′ a′ v	cat: sn′ snp	cat: sn
sem: pl exh	sem: sg	sem: pres	sem: indef sg	sem: sg
mdr:	mdr:	mdr:	mdr:	mdr:
fnc:	fnc:	arg:	fnc:	fnc:
idy:	idy:	prn:	idy:	idy:
prn:	prn:		prn:	prn:

Each word is analyzed as a feature structure[2] called *proplet*. A feature structure is defined as a set of features. A feature is defined as an attribute-value pair. As feature structures, proplets are special only insofar as the values of attributes may not be feature structures. In

other words, proplets are 'flat' or 'non-recursive' feature structures. For better readability, the order of attributes in a proplet is fixed. They have the following interpretation:

2.2 DEFINITION OF PROPLET ATTRIBUTES

sur	= surface
noun, verb, adj	= core attributes, represent the part of speech
cat	= category, specifies combinatorial properties
sem	= semantics, specifies non-combinatorial properties
mdr	= modifier modifying a noun, a verb, or an adjective
mdd	= modified, i.e. the noun, verb, or adjective which a modifier modifies
fnc	= functor, specifies the verb belonging to a noun
arg	= argument, specifies the nouns belonging to a verb
idy	= identity, indicates whether two nouns are distinct or not
prn	= proposition number, holding the proplets of a proposition together

cat and sem are called the grammatical attributes, fnc, arg, and mdd the obligatory continuation attributes, mdr the optional continuation attribute, and idy and prn the book-keeping attributes. In a complete result, obligatory attributes must have a non-NIL value, while optional attributes may have the value NIL. Continuation attributes specify the primary grammatical relations between proplets, namely the functor-argument structure and coordination. The book-keeping attributes have integers as values which are incremented by the control structure of the parser each time a new value is provided.

Next consider the proplet values needed for a complete treatment of internal and external agreement in English:

2.3 DEFINITION OF PROPLET VALUES

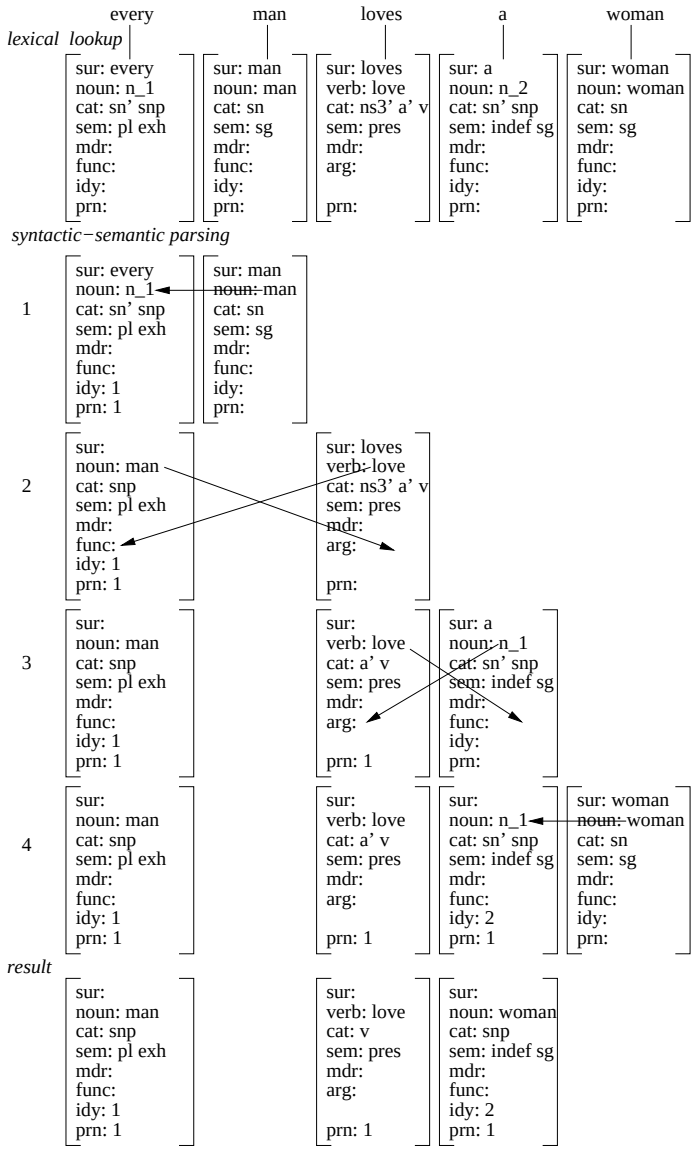
value:	*explanation:*	*attribute:*
n′	unrestricted nominative valency position (slept)	cat of a verb
ns1′	nominative singular 1. person (am)	cat of a verb
n-s3′	nominative except singular 3. person (sleep)	cat of a verb
ns3′	nominative singular 3. person (sleeps)	cat of a verb
n-s13′	nominative except singular 1. and 3. person (are)	cat of a verb
ns13′	nominative singular 1. and 3. person (was)	cat of a verb
d′	dative valency position (gave)	cat of a verb
a′	accusative valency position (kiss)	cat of a verb
nm	proper name (John)	cat of a noun
ns1	nominative singular 1. person (I)	cat of a (pro)noun
ns3	nominative singular 3. person (he)	cat of a noun
pro2	nominative and oblique 2. person sg and pl filler (you)	cat of a (pro)noun
np-2	nominative plural 1. and 3. person (we, they)	cat of a noun
snp	singular noun phrase (the girl)	cat of a noun
pnp	plural noun phrase (the girls)	cat of a noun
obq	oblique (non-nominative) (me, him, her, us, them)	cat of a (pro)noun
nn′	noun valency position unmarked for number (the)	cat of a determiner
pn′	noun valency marked for plural (all)	cat of a determiner
sn′	noun valency marked for singular (every)	cat of a determiner

[2] Feature structures were first introduced by M. Minsky 1974.

value:	*explanation:*	*attribute:*
nn	noun valency filler unmarked for number (sheep)	cat of a noun
pn	noun valency filler for plural (books)	cat of a noun
sn	noun valency filler for singular (book)	cat of a noun
def	definite	sem of a noun
exh	exhaustive	sem of a noun
f	femininum	sem of a noun
indef	indefinite	sem of a noun
m	masculinum	sem of a noun
pl	plural	sem of a noun
sg	singular	sem of a noun
sel	selective	sem of a noun

Having defined the attributes and the values used in 2.1, these lexical proplets must be connected. This process is shown schematically in the following time-linear derivation.

2.4 TIME-LINEAR DERIVATION OF Every man loves a woman



In line 1, the substitution value n_1 in the core attribute of *every* is replaced by the core value of the second proplet *man*, which is then discarded. In line 2, the new core value man of the

first proplet is copied into the arg slot of the third proplet, and the core value love is copied into the fnc slot of the first proplet, establishing a functor-argument relation between the two proplets. In line 3, the substitution value n_2 serving as the core value of the indefinite article is copied into the arg slot of the *love* proplet, and the core value love is copied into the fnc slot of the definite article, establishing a second functor-argument relation. Finally, the substitution value n_2 is replaced by the core value woman of the last proplet, which is then discarded.

The rules of the derivation are based on matching between proplet patterns of the rule level and proplets of the language level. Consider the rule application in the first composition:

2.5 EXAMPLE OF AN **LA-hear** RULE APPLICATION

rule name	*ss pattern*	*nw pattern*	*operations*	*rule package*	
2 DET+NN	[noun: N_*n* cat: N′ X]	[noun: α cat: N]	delete N′ ss.cat replace α N_*n* copy$_{ss}$	{6 NOM+FV, ...}	*rule level*

[sur: every noun: n_1 cat: sn′ snp sem: pl exh mdr: fnc: idy: 1 prn: 1]	[sur: man noun: *man* cat: sn sem: sg mdr: fnc: idy: prn:]	⟹	[sur: noun: *man* cat: snp sem: pl exh mdr: fnc: idy: 1 prn: 1]		*language level*

The *rule level* consists of (i) a rule name, (ii) a pattern for the sentence start (*ss*), (iii) a pattern for the next word (*nw*), (iv) a set of operations, and (v) a rule package. The rule patterns are matched with the proplets at the *language level*, whereby the following conditions apply.

2.6 MATCHING CONDITIONS

1. *Attribute condition*
   The matching between two proplets A and B requires that the intersection of their attributes contains a predefined list of attributes:

   $$\{\text{list}\} \subseteq \{\{\text{proplet-A-attributes}\} \cap \{\text{proplet-B-attributes}\}\}$$

2. *Value condition*
   The matching between two proplets requires that the variables (and a fortiori the constants) of their common attributes are compatible.

In 2.5, the attributes of the rule patterns are a subset of the attributes of the language proplets.

The rule patterns in 2.5 use the substitution variable N_*n*, which is restricted to the substitution values n_1, n_2, etc, the binding variable α, which is restricted to single core values, the binding variables N′ and N, the restrictions of which are defined in 2.8, and the unrestricted binding variable X, which matches any sequence of length zero to four.

During matching, the variables of the rule level are bound to corresponding values at the language level. For example, N′ is bound to sn′ and X is bound to snp. This is the basis for executing the rule level operations at the language level. Variable binding based on matching is called *vertical binding*, in contrast to the *horizontal* variable binding based on quantifiers, as in Predicate Calculus (cf. Section 1).

The first operation, delete N′ ss.cat, cancels the category segment sn′, representing a valency position for a singular noun, in the cat slot of the determiner. The second operation, replace α N_*n*, replaces the substitution value n_1 in the noun slot of the determiner with man. The third operation, copy$_{ss}$, specifies that only the first proplet, i.e. the sentence start, is in the result, while the second proplet, i.e. the next word, is discarded.

The operations can only apply if the matching is successful. For this, the binding of the variables must fulfill (i) the restrictions on variables and (ii) the agreement conditions.

2.7 RESTRICTIONS OF VARIABLES

N	ε	{nn, sn, pn}	fillers for a noun valency slot
N′	ε	{nn′, sn′, pn′}	valency slots for a noun in a determiner
NP	ε	{pro2, nm, ns1, ns3, np-2, snp, pnp, pn, obq}	fillers for a noun phrase valency slot
NP′	ε	{n′, n-s3′, ns1′, ns3′, ns13′, n-s13′, d′, a′}	valency slots for a noun phrase in a verb

In example 2.5, the value corresponding to the restricted variable N′ is sn′, while the value corresponding to the restricted variable N is sn. Given that these two values are in the respective restriction sets of the two variables, binding can proceed.

The internal agreement and external agreement of nouns in English is handled by the following definition:

2.8 AGREEMENT CONDITIONS

if N′ = nn′, then N ε {nn, sn, pn}
if N′= sn′, then N ε {nn, sn}
if N′= pn′, then N ε {nn, pn}

if NP = ns1, then NP′ ε {n′, n-s3′, ns1′, ns13′}
if NP = pro2, then NP′ ε {n′, n-s3′, n-s13′, d′, a′}
if NP = ns3, then NP′ ε {n′, ns3′, ns13′}
if NP = np-2, then NP′ ε {n′, n-s3′, n-s13′}
if NP ε {nm, snp}, then NP′ ε {n′, ns3′, ns13′, d′, a′}
if NP = pnp, then NP′ ε {n′, n-s3′, n-s13′, d′, a′}
if NP = obq, then NP′ ε {d′, a′}

In example 2.5, the second condition of the above definition is fulfilled. Therefore, matching is successful, the operations apply, a new sentence start is derived, a new next word is looked up from the lexicon, and the rules of the package are applied to the new sentence start and the new next word.

3 Comparing the Two Approaches

In Predicate Calculus, the sentence derived in the previous section is represented as follows:

3.1 PC ANALYSIS OF Every man loves a woman

Reading 1: ∀x [man(x) → ∃y [woman(y) & love(x,y)]
Reading 2: ∃y [woman(y) & ∀x [man(x) → love(x,y)]

On reading 1, it holds for every man that there is some woman who he loves. On reading 2, there is a certain woman, e.g. Marilyn Monroe, who is loved by every man.

The two formulas of Predicate Calculus are based on the notions of functor-argument structure, coordination, and coreference, though in a manner different from their use in Database

Semantics. Functor-argument structure is used in man(x), woman(y), and love(x,y), coordination is used in [man(x) → P] and [woman(y) & Q], and coreference is expressed by the quantifiers and the horizontally bound variables in ∀x [man(x) ... love(x,y)] and ∃y [woman(y)... love(x,y)].

The meanings of every and a, handled in PC at the highest level of logical syntax using quantifiers, variables, and connectives, is expressed in DBS by values of the sem attribute:

3.2 RESULT OF PARSING Every man loves a woman IN DBS

sur:	sur:	sur:
noun: *man*	verb: *love*	noun: *woman*
cat: snp	cat: v	cat: snp
sem: pl exh	sem: pres	sem: indef sg
mdr:	mdr:	mdr:
fnc: love	arg: man woman	fnc: love
idy: 1	prn: 1	idy: 2
prn: 1		prn: 1

In this analysis, the sentence is not ambiguous: it has only reading 1 of 3.1 – which is entailed by reading 2 (i.e. reading 1 is true, whenever reading 2 is true, but not vice versa). The intuition represented by the two PC readings is treated as matter of pragmatic interpretation.

Furthermore, the DBS analysis uses only intra-propositional functor-argument structure; as in the natural surface, there is neither coordination nor coreference. Treated as determiners, every and a are each fused with their noun into a single proplet (function word absorption).

The meanings of the two determiners are expressed by the values of their respective sem attribute, which are pl exh (for plural exhaustive) in the case of every and sg sel (for singular selective) in the case of a. Before we turn to the set-theoretic intuitions underlying these values, let us consider the treatment of the definite article in the two approaches.

3.3 PC ANALYSIS OF The girl sleeps

$$\exists x\,[\forall y\,[\mathrm{girl}(x) \rightarrow x = y]\ \&\ \mathrm{sleep}(y)]$$

Again, the meaning of the determiner is expressed at the highest level of the logical syntax. According to Russell 1905, the salient semantic property of the definite article is uniqueness, coded by the uniqueness condition ∀y [girl(x) → x = y]. The formula is true relative to a model if there exists an x, such that it holds for all y, if x is a girl, then x = y and y sleeps.

The DBS analysis, in contrast, consists of two proplets in a functor-argument relation. The contribution of the definite article is represented by the value def of the noun's sem attribute.

3.4 DBS ANALYSIS OF The girl is sleeping

sur:	sur:
noun: girl	verb: sleep
fnc: sleep	arg: girl
cat: snp	cat: v
sem: def sg	sem: pres prog
mdr:	mdr:
idy: 3	prn: 3
prn: 2	

There are also plural definite noun phrases, characterized by the feature [sem: def pl].

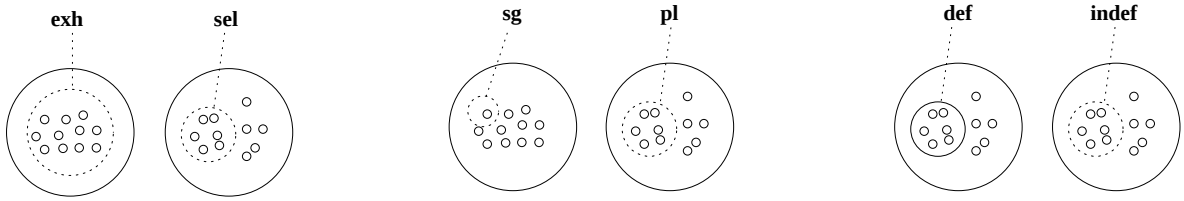
The values exh (exhaustive), sel (selective), sg (singular), pl (plural), def (definite), and indef (indefinite) are used in different combinations to characterize the following kinds of noun phrases in English:

3.5 THE sem VALUES OF DIFFERENT DETERMINER-NOUN COMBINATIONS

a girl	[sem: indef sg]
some girls	[sem: indef pl sel]
all girls	[sem: pl exh]
the girl	[sem: def sg]
the girls	[sem: def pl]

Consider the following proposal to interpret these values set-theoretically:

3.6 SET-THEORETIC INTERPRETATION OF exh, sel, sg, pl, def, indef

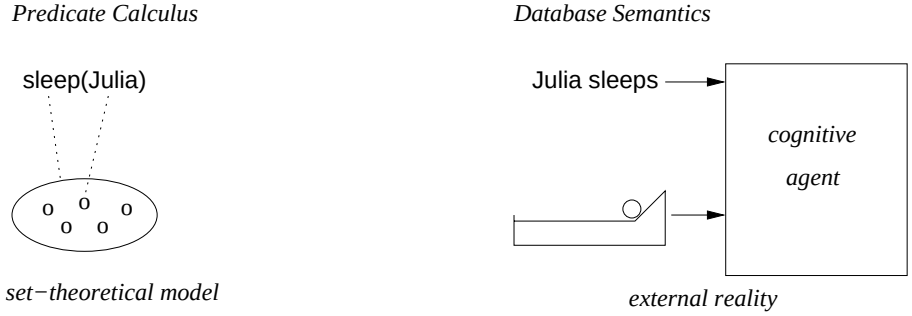


The value exh refers to all members of a set, called the domain, while sel refers only to some. The value sg refers to a single member of the domain, while pl refers to more than one. The value def refers to a prespecified subset, while no such subset is presumed by indef.

Each value can only be combined with values from the other pairs. Thus exh canot be combined with sel, sg cannot be combined with pl, and def cannot be combined with indef. However, the combinations exh pl, sel pl, indef pl, indef sg, def sg and indef sg are legitimate and have different meanings. The combination exh sg is theoretically possible, but makes little sense because the domain would have to be a unit set.

The DBS approach differs from PC in that PC uses the words *some* and *all* in the meta-language to define the words some and all in the object-language (as shown by the use of the variable assignment function g' described in Section 1), while DBS is based on a procedural interpretation. This difference is based on profoundly different ontological assumptions of the two approaches, illustrated below with the most simple sentence Julia sleeps.

3.7 REFERENCE IN THE ALTERNATIVE ONTOLOGIES OF PC AND DBS



In PC, the sentence is formalized as *sleep (Julia)*, indicating that *sleep* is a functor denoting a set while *Julia* is an argument denoting an element. The meta-language-defined relation between the language expressions and their set-theoretic denotations is indicated by the dotted lines. There is no agent and therefore there are no external interfaces. Thus there is neither a need nor a place for agent-internal procedures.

The DBS example on the right shows an agent in the hearer-mode. The agent relates the sentence Julia sleeps to the referent solely by means of cognition. The agent, the language expression, and the referent with its property of sleeping are all part of the real world. The agent interacts with the real world based on its external interfaces for action.

Regarding the interpretation of determiners, consider a robot in the speaker-mode. If it perceives the set-theoretic situation corresponding to exh and pl as shown in 3.6, it will use the determiner all, and similarly with the other values. Correspondingly, if a robot in the hearer-mode hears the noun phrase all girls, for example, it will be able to draw the corresponding set-theoretic situation or choose the right schema from several alternatives.

A related difference between PC and DBS is that the semantics of PC is based on truth-conditions, while that of DBS is not. Instead, DBS handles truth as procedural assertions. For example, if a robot observes correctly that every girl is sleeping and communicates this fact by saying every girl is sleeping, it is speaking truely. Semantically, every girl is sleeping asserts that there is a set of more than one girl and all elements of the set participate in whatever is asserted by the verb.

4 Korean Noun Phrases with Numerals and Classifiers

Having shown how quantification in English is handled in Database Semantics and how it compares with quantification in Predicate Calculus, we now extend the descriptive coverage to the noun phrases of Korean, a language unrelated to English and of a different type (cf. Chang 1996). For each kind of Korean NP, an automatic derivation and representation in Database Semantics will be provided.

Our discussion of the internal structure of noun phrases in Korean begins with constructions involving numerals and classifiers. Consider the most simple kind of NP in Korean, namely a noun phrase consisting of a single common noun *sonye* '`girl`'.

4.1 TIME-LINEAR DERIVATION OF sonye-ka canta '*(A) girl sleeps.*'

`sonye-ka  canta`

lexical lookup

sur: sonye-ka	sur: canta
noun: girl	verb: sleep
cat: (n)	cat: (nom' v)
case: (nom)	sem: (present)
sem: ()	arg: ()
mdr: ()	ctn: (0 nil)
fnc: @2	

syntactic-semantic parsing

```
1.1
NOUN+FV     copy(NW)     {IP+PM}
            acopy(SS1 NWL1)
            ecopy(NW1 FNC)
            cancel(NP')
[noun: $SS1    ]  [verb: $NW1     ]
[cat : ($X $NP)]  [cat : ($NP' v) ]
[case: $CS     ]  [sem : ($NW2)   ]
[sem : $SSL1   ]  [arg : $NWL1    ]
[fnc : $FNC    ]

[sur : sonye-ka]  [sur : canta    ]
[noun: girl    ]  [verb: sleep    ]
[cat : (n)     ]  [cat : (nom' v) ]
```

```
[case: (nom)   ]  [sem : (present)]
[sem : ()      ]  [arg : ()       ]
[mdr : ()      ]  [ctn : (0 nil)  ]
[fnc : @2      ]  [wrd : 2        ]
[wrd : 1       ]  [prn : (1)      ]
[prn : (1)     ]

+++ Final State +++

[sur : sonye-ka]  [sur : canta    ]
[noun: girl    ]  [verb: sleep    ]
[cat : (n)     ]  [cat : (v)      ]
[case: (nom)   ]  [sem : (present)]
[sem : ()      ]  [arg : (girl)   ]
[mdr : ()      ]  [ctn : (0 nil)  ]
[fnc : sleep   ]  [wrd : 2        ]
[wrd : 1       ]  [prn : (1)      ]
[prn : (1)     ]
```

The above derivation[3] is rather trivial in Database Semantics: the key operation in line 1 is to copy the core value sleep of the second proplet into the fnc slot of the first proplet, and to copy the core value girl into the arg slot of the second proplet, thus establishing a functor-argument relation between the two. Also the nom′ element of the cat value of the second proplet is now saturated and deleted due to the presence of the first proplet as its argument.

One thing to note in the lexical entry of the noun *sonye-ka* 'girl-NOM' is that a new attribute case is introduced to keep the information that originates from the case particle, namely, the nominative case marker *-ka* in the above example. The primary candidate for the attribute to hold the case information would be the cat attribute as it typically handles 'subcategorization' aspect of the grammar, but we tentatively separate it from the rest of the cat value in this paper due to the reason to be discussed shortly.

There are various ways a noun phrase can represent its quantity related information on the surface in Korean. The most typical way is to make use of numeral classifiers. For example, *a girl/one girl* in English can be represented as follows.

4.2 POST-NOMINAL NUMERALS AND CLASSIFIERS: NOUN-NUM-CL

1. *sonye han myeng*

   girl a/one CL

The classifier *myeng* is used for nouns that refer to human beings or its hyponyms, and its semantic role is to 'unitize' the referents of the noun (cf. Chae 1996). One can say its semantic content is somewhat redundant, as is shown by its optionality. With or without the classifier *myeng*, the meaning of the noun phrase remains the same.

1. *sonye hana*

   girl a/one

In the case of some numerals, as in the one above, there is a change in their form, namely from the adnominal form to the nominal one, when the classifiers are absent. For example, the adnominal form *han* 'one' has its nominal counterpart *hana*, which is appropriate when there is no classifier immediately following, or when it is the sole member of the whole NP.

[3] All constructions shown here have been implemented in Java, and an actual output of the implemented grammar will be shown hereafter with some minor editing to delete some irrelevant portions of the derivation.

An interesting complication to this construction[4] is that some case-markers like the nominative *-ka(/-i)* and accusative *-lul(/-ul)* can attach either to the host noun or to the classifier,[5] or to both. Given that the classifier itself is optional, there would be up to six possible variants of the construction.[6]

1. *sonye-ka han myeng*
   `girl-NOM one CL`
2. *sonye han myeng-i*
   `girl one CL-NOM`
3. *sonye-ka han myeng-i*
   `girl-NOM one CL-NOM`
4. *sonye-ka hana*
   `girl-NOM a/one`
5. *sonye hana-ka*
   `girl a/one-NOM`
6. *sonye-ka hana-ka*
   `girl-NOM a/one`

Then the eight possible variations can be summarized as follows:

4.3 PATTERNS OF POST-NOMINAL NUMERALS AND CLASSIFIERS

1. NOUN(*case*) NUM CL(*case*)
2. NOUN(*case*) NUM(*case*)

Note that the case marker can attach virtually to any element in the NP structure. Whichever element the case marker attaches to, its value should be copied to the proplet that remains to be included in the resulting representation. If we were to keep the information as part of the cat value, then linking it to the surface position of the case marker would get complicated, and that is why we introduce a separate attribute case. Since the case marker can attach to NOUN, CL, or NUM when it appears in a nominal position, all their proplets will have the case attribute in the lexicon.

Now the question is how we can deal with these various Korean quantificational constructions in the framework of Database Semantics. The first issue would be whether we should

[4]There are some issues concerning the constituency of the above expressions, that is, whether each expression forms a single constituent or not at the syntactic level, as the second part of the expression, namely, numeral + classifier *han myeng* can sometimes 'float' from its host noun *sonye-ka* thus allowing an intervening element in between. We will not touch on the issue here, but for further discussion see Sohn 1999, Kang 2002, and references cited therein.

[5]Another point to note is that while distribution and use of classifiers are largely the same in Korean and Japanese, one major difference is that in Japanese case markers cannot attach to the classifiers (cf. Miyagawa 1989).

[6]Yet another optionality, though not to be discussed in this paper, is attested by the plural marker *-tul*. In the examples given above, *-tul* can optionally attach to the noun *sonye* '`girl`', thus making the total number of variants twelve.

treat classifiers as constituting a separate proplet apart from its host noun in the resulting semantic representation. In this paper, we will assume that information contained in the classifier is copied to its preceding numeral to form a single proplet. This is motivated by the observation that the classifier is optional and distributionally dependent on its adjacent numeral. Typical classifiers cannot stand alone without a preceding numeral (Im 2002).[7]

We will show next how the sequence NOUN-NUM-CL sequence is derived step by step in Database Semantics, starting from an input sentence, ending up with a set of connected proplets. The relevant grammar is presented in Section 7.

4.4 DERIVATION OF sonye-ka han myeng canta *'One girl sleeps.'*

```
sonye-ka      han      myeng      canta
```

lexical lookup

sur: sonye-ka	sur: han	sur: myeng	sur: canta
noun: girl	noun: @1	noun: @1	verb: sleep
cat: (n)	cat: (cl′ n′ n)	cat: (cl)	cat: (nom′ v)
case: (nom)	case: ()	case: ()	sem: (present)
sem: ()	sem: (one)	sem: (individual)	arg: ()
mdr: ()	mdr: ()		ctn: (0 nil)
fnc: @2	fnc: nil		

Syntactic-semantic parsing

```
1.1
NOUN+NUM     copy(NW)     {NUM+CL NOUN+FV}
             ecopy(SS.sur NW.sur)
             ecopy(SS.noun NW.noun)
             cancel(N')
             ecopy(SS.case NW.case)
             acopy(SS.sem NW.sem)
             acopy(SS.mdr NW.mdr)
             ecopy(SS.fnc NW.fnc)
[noun: $SS1    ]  [noun: $NW1      ]
[cat : ($N)    ]  [cat : ($X $N' n)]

[sur : sonye-ka]  [sur : han       ]
[noun: girl    ]  [noun: @1        ]
[cat : (n)     ]  [cat : (cl' n' n)]
[case: (nom)   ]  [case: ()        ]
[sem : ()      ]  [sem : (one)     ]
[mdr : ()      ]  [mdr : ()        ]
[fnc : @2      ]  [fnc : nil       ]
[wrd : 1       ]  [wrd : 2         ]
[prn : (1)     ]  [prn : (1)       ]

2.11
NUM+CL     acopy(NWL1 SSL1)     {NOUN+FV NUM+NOUN}
           cancel(CL')
[noun: $SS1           ]  [noun: $NW1        ]
[cat : ($X $CL' $Y $N)]  [cat : ($CL)       ]
[case: $SSL2          ]  [case: $NWL2       ]
[sem : $SSL1          ]  [sem : $NWL1       ]
```

[7] Most classifiers do not allow any modifiers other than the numerals.

i) *sonye han myeng*
girl one CL
ii) **sonye yeyppun myeng*
girl pretty CL
iii) **sonye motun myeng*
girl all CL

```
[sur : sonye-ka       ]  [sur : myeng       ]
[noun: girl           ]  [noun: @1          ]
[cat : (cl' n)        ]  [cat : (cl)        ]
[case: (nom)          ]  [case: ()          ]
[sem : (one ())       ]  [sem : (individual)]
[mdr : (())           ]  [wrd : 3           ]
[fnc : @2             ]  [prn : (1)         ]
[wrd : 2              ]
[prn : (1)            ]

3.111
NOUN+FV     copy(NW)     {IP+PM}
            acopy(SS1 NWL1)
            ecopy(NW1 FNC)
            cancel(NP')
[noun: $SS1    ]  [verb: $NW1     ]
[cat : ($X $NP)]  [cat : ($NP' v) ]
[case: $CS     ]  [sem : ($NW2)   ]
[sem : $SSL1   ]  [arg : $NWL1    ]
[fnc : $FNC    ]

[sur : sonye-ka]  [sur : canta    ]
[noun: girl    ]  [verb: sleep    ]
[cat : (n)     ]  [cat : (nom' v) ]
[case: (nom)   ]  [sem : (present)]
[sem : ()      ]  [arg : ()       ]
[mdr : ()      ]  [ctn : (0 nil)  ]
[fnc : @2      ]  [wrd : 4        ]
[wrd : 1       ]  [prn : (1)      ]
[prn : (1)     ]

+++ Final State +++

[sur : sonye-ka             ]  [sur : canta    ]
[noun: girl                 ]  [verb: sleep    ]
[cat : (n)                  ]  [cat : (v)      ]
[case: (nom)                ]  [sem : (present)]
[sem : (one () (individual))]  [arg : (girl)   ]
[mdr : (())                 ]  [ctn : (0 nil)  ]
[fnc : sleep                ]  [wrd : 4        ]
[wrd : 2                    ]  [prn : (1)      ]
[prn : (1)                  ]
```

In line 1, where the noun-numeral input sequence matches the rule NOUN+NUM, the relevant values of the noun girl, including the case value nom, are all copied to the numeral one proplet including the core value noun, and the girl proplet is then discarded. Note that the cat value of the one proplet remains in place except for the N′ element which is cancelled as it is saturated by the preceding noun.

In line 2, relevant information is copied from the new next word, the classifier *myeng*, to the sentence start resulting from the previous composition. The key value to be copied is the sem value, individual, which is added to the sem value of the sentence start. Here the relevant element of the cat value is also satuarated due to the presence of the classifier *myeng* and is thus cancelled. The rest of the parsing is the same as before, the completed noun phrase combining with the verb.

One issue that has to be addressed concerns appropriate percolation of the case value. Since the case information of the noun will eventually survive to become the case value of the whole NP, there has to be a way to keep the case information when the case marker is attached only to the classifier but not to the noun, as in the pattern *sonye han myeng-i* `girl one CL-NOM`. One solution is to introduce an extra set of rules to handle these cases. Another solution would be to make the ecopy operation in the rule conditional to the presence or absence of the value of the case attribute: When the classifier becomes the next word in the derivation process, the case value of the classifier will be copied as the case value of the sentence start

provided that its value is null.[8]

Returnting to the possible patterns of quantificational NPs in Korean, even the noun is not always required in an NP. For example, an NP may consist of a NUM only.

4.5 RESULT OF PARSING hana-ka canta *'One sleeps.'*

sur : hana-ka	sur : canta
noun: @1	verb: sleep cat : (v)
cat : (n′ n)	sem : (present)
case: (nom)	arg : (@1)
mdr : ()	ctn : (0 nil)
sem : (one)	wrd : 2
fnc : sleep	prn : (1)
wrd : 1	
prn : (1)	

The sentence does not seem specific enough to be counted as an independent proposition, but there is no doubt that one comes across such kind of utterances rather frequently in everyday speech. We can assume that the rest of the information to make the sentence concrete enough will come from the discourse, and the unsaturated element of the cat value of the one proplet will eventually be cancelled or linked to some salient discourse referent through inference.

5 Prenominal Numerals in Korean

So far we have considered cases of quantificational NP in Korean where the numeral follows the related noun. But the numerals can precede the noun as well – with or without an intervening classifier.

5.1 PRENOMINAL NUMERALS AND CLASSIFIERS: NUM-CL-NOUN

1. *han myeng sonye*

   `one CL girl`

2. *han sonye*

   `one girl`

When the classifier precedes the noun, its case relation to the noun becomes more restricted and allows only a genitive case marker as its suffix.

1. *han myeng-ui sonye*

   `one CL-GEN girl`

Therefore, there will be all six variants of the prenominal numeral/classifier construction for noun phrases in Korean.

[8] However, allowing a rule operation to be sensitive to the context might cause some undesirable complication in the whole system, so this would obviously require some further investigation.

5.2 PATTERNS OF PRENOMINAL NUMERALS AND CLASSIFIERS

1. NUM CL(*case*$_{gen}$) NOUN(*case*)
2. NUM NOUN(*case*)

For one of these new constructions, namely the NUM-CL-NOUN sequence, we provide the following derivation in Database Semantics.

5.3 DERIVATION OF han myeng-ui sonye-ka canta '*One girl sleeps.*'

```
han     myeng-ui    sonye-ka        canta
```

lexical lookup

sur: han noun: @1 cat: (cl′ n′ n) case: () sem: (one) mdr: () fnc: nil	sur: myeng-ui noun: @1 cat: (cl) case: (gen) sem: (individual)	sur: sonye-ka noun: girl cat: (n) case: (nom) sem: () mdr: () fnc: @2	sur: canta verb: sleep cat: (nom′ v) sem: (present) arg: () ctn: (0 nil)

Syntactic-semantic parsing

```
1.1
NUM+CL      acopy(NWL1 SSL1)      {NOUN+FV NUM+NOUN}
            cancel(CL')
[noun: $SS1            ]  [noun: $NW1        ]
[cat : ($X $CL' $Y $N)]  [cat : ($CL)       ]
[case: $SSL2           ]  [case: $NWL2       ]
[sem : $SSL1           ]  [sem : $NWL1       ]

[sur : han             ]  [sur : myeng-ui    ]
[noun: @1              ]  [noun: @1          ]
[cat : (cl' n' n)      ]  [cat : (cl)        ]
[case: ()              ]  [case: (gen)       ]
[sem : (one)           ]  [sem : (individual)]
[mdr : ()              ]  [wrd : 2           ]
[fnc : nil             ]  [prn : (1)         ]
[wrd : 1               ]
[prn : (1)             ]

2.12
NUM+NOUN      ecopy(NW.sur SS.sur)      {NOUN+FV}
              ecopy(NW.noun SS.noun)
              cancel(N')
              acopy(NW.case SS.case)
              acopy(NW.sem SS.sem)
              acopy(NW.mdr SS.mdr)
              ecopy(NW.fnc SS.fnc)
[noun: $SS1               ]  [noun: $NW1    ]
[cat : ($X $N' $Y n)      ]  [cat : ($N)    ]
[sem : $SSL1              ]

[sur : han                ]  [sur : sonye-ka]
[noun: @1                 ]  [noun: girl    ]
[cat : (n' n)             ]  [cat : (n)     ]
[case: ()                 ]  [case: (nom)   ]
[sem : (one (individual))]  [sem : ()      ]
[mdr : ()                 ]  [mdr : ()      ]
[fnc : nil                ]  [fnc : @2      ]
[wrd : 1                  ]  [wrd : 3       ]
[prn : (1)                ]  [prn : (1)     ]

3.121
```

```
NOUN+FV     copy(NW)     {IP+PM}
            acopy(SS1 NWL1)
            ecopy(NW1 FNC)
            cancel(NP')
[noun: $SS1                 ]  [verb: $NW1     ]
[cat : ($X $NP)             ]  [cat : ($NP' v) ]
[case: $CS                  ]  [sem : ($NW2)   ]
[sem : $SSL1                ]  [arg : $NWL1    ]
[fnc : $FNC                 ]

[sur : sonye-ka             ]  [sur : canta    ]
[noun: girl                 ]  [verb: sleep    ]
[cat : (n)                  ]  [cat : (nom' v) ]
[case: ((nom))              ]  [sem : (present)]
[sem : (one (individual) ())]  [arg : ()       ]
[mdr : (())                 ]  [ctn : (0 nil)  ]
[fnc : @2                   ]  [wrd : 4        ]
[wrd : 1                    ]  [prn : (1)      ]
[prn : (1)                  ]

+++ Final State +++

[sur : sonye-ka             ]  [sur : canta    ]
[noun: girl                 ]  [verb: sleep    ]
[cat : (n)                  ]  [cat : (v)      ]
[case: ((nom))              ]  [sem : (present)]
[sem : (one (individual) ())]  [arg : (girl)   ]
[mdr : (())                 ]  [ctn : (0 nil)  ]
[fnc : sleep                ]  [wrd : 4        ]
[wrd : 1                    ]  [prn : (1)      ]
[prn : (1)                  ]
```

In line 1, the sem value of the classifier proplet is copied to the preceding numeral proplet, while the value cl' of the cat attribute of the sentence start gets cancelled due to the presence of the following classifier. The numeral proplet then receives again some relevant values from the noun girl proplet, and the proplet of the next word is discarded. The numeral proplet, having collected all the relevant information from the classifier and the noun, then gets combined with the verb, saturating the nom' value of the verb proplet's cat attribute.

Now we turn to noun phrase quantification phenomena that involve words other than numerals. As a typical example of those words, we consider the universal quantifier.

6 Universal Quantification in Korean

For universal quantification in Korean, there are again two possibilities concerning the position of the universal expression, either to the right of the related noun or to its left, with appropriate morphological adjustments. However, classifiers are not appropriate in either case as they can co-occur with numerals only.

1. *sonye motwu*

   `girl all`

2. *motun sonye*

   `all girl`

In the case of a post-nominal quantifier, it can host a case marker like numerals in the above. Then there would be four possible variants for 1, and two for 2. They are summarized in the following.

6.1 PATTERNS OF QUANTIFIERS

1. NOUN(case) Q(case)

2. Q NOUN(case)

However, unlike the universal quantifier, the selective quantifier *myech* '`some`' behaves in the same manner as the numerals, combining with the classifiers, so it will be treated as a kind of numeral.

Let us now consider another derivation, which involves universal quantification.

6.2 DERIVATION OF motun sonye-ka canta ' *Every girl sleeps.*'

`motun sonye-ka canta`

lexical lookup

sur: motun	sur: sonye-la	sur: canta
noun: @1	noun: girl	verb: sleep
cat: (n′ n)	cat: (n)	cat: (nom′ v)
case: ()	case: (nom)	sem: (present)
sem: (pl exh)	sem: ()	arg: ()
mdr: ()	mdr: ()	ctn: (0 nil)
fnc: nil	fnc: @2	

Syntactic-semantic parsing

```
1.1
NUM+NOUN     ecopy(NW.sur SS.sur)     {NOUN+FV}
             ecopy(NW.noun SS.noun)
             cancel(N')
             acopy(NW.case SS.case)
             acopy(NW.sem SS.sem)
             acopy(NW.mdr SS.mdr)
             ecopy(NW.fnc SS.fnc)
[noun: $SS1         ]  [noun: $NW1    ]
[cat : ($X $N' $Y n)]  [cat : ($N)    ]
[sem : $SSL1        ]

[sur : motun        ]  [sur : sonye-ka]
[noun: girl         ]  [noun: girl    ]
[cat : (n' n)       ]  [cat : (n)     ]
[case: ()           ]  [case: (nom)   ]
[sem : (pl exh)     ]  [sem : ()      ]
[mdr : ()           ]  [mdr : ()      ]
[fnc : nil          ]  [fnc : @2      ]
[wrd : 1            ]  [wrd : 2       ]
[prn : (1)          ]  [prn : (1)     ]

2.11
NOUN+FV     copy(NW)     {IP+PM}
            acopy(SS1 NWL1)
            ecopy(NW1 FNC)
            cancel(NP')
[noun: $SS1       ]  [verb: $NW1     ]
[cat : ($X $NP)   ]  [cat : ($NP' v) ]
[case: $CS        ]  [sem : ($NW2)   ]
[sem : $SSL1      ]  [arg : $NWL1    ]
[fnc : $FNC       ]

[sur : sonye-ka   ]  [sur : canta    ]
[noun: girl       ]  [verb: sleep    ]
[cat : (n)        ]  [cat : (nom' v) ]
[case: ((nom))    ]  [sem : (present)]
[sem : (pl exh ())]  [arg : ()       ]
[mdr : (())       ]  [ctn : (0 nil)  ]
[fnc : sleep      ]  [wrd : 3        ]
```

```
[wrd : 1          ]  [prn : (1)      ]
[prn : (1)        ]

+++ Final State +++

[sur : sonye-ka   ]  [sur : canta    ]
[noun: girl       ]  [verb: sleep    ]
[cat : (n)        ]  [cat : (v)      ]
[case: ((nom))    ]  [sem : (present)]
[sem : (pl exh ())]  [arg : (girl)   ]
[mdr : (())       ]  [ctn : (0 nil)  ]
[fnc : sleep      ]  [wrd : 3        ]
[wrd : 1          ]  [prn : (1)      ]
[prn : (1)        ]
```

The word *motun* 'every' has the sem values pl and exh as part of its lexical information (See Section 3.6). The quantificational determiner collects relevant information from its adjacent noun, and then gets connected to the verb, forming a set of connected proplets. The derivation in the above is just like the ones for numerals of similar structure.

7 Definition of LA-Korean.1

So far we have considered 28 patterns of the noun phrase internal stuctures in Korean with respect to numerals, classifiers and quantifiers. We have also provided some key derivations showing how we can parse the noun phrases in Korean step-by-step in Database Semantics. In conclusion let us present the grammar package, **LA-Korean.1**, on which the automatic parsing of the derivations has been based.

7.1 RESTRICTIONS OF VARIABLES

N ε {n}
NP ε {n nom acc dat}
N′ ε {n′}
NP′ ε {n′ nom′ acc′ dat′}
CL ε {cl}
CL′ ε {cl′}
CS′ ε {nom acc dat}

The second and the fourth definitions are needed to handle the matching between a non-case marked noun phrase and its verb. The last definition can be used to restrict the distribution of case information.

7.2 AGREEMENT CONDITIONS

if CL′ ε {cl′}, then CL ε {cl}
if NP ε {nom n}, then NP′ ε {nom′}
if NP ε {acc n}, then NP′ ε {acc′}
if NP ε {dat n}, then NP′ ε {dat′}

The first definition would be needed to distinguish between quantificational expressions which allow classifiers, and other quantifiers which do not. It blocks, for example, a Q-CL sequence from being parsed.

7.3 DEFINITION OF START STATE, RULES, AND FINAL STATES

ST$_S$ =$_{def}$ { ([cat: X] {1 NUM+CL 2 NUM+NOUN 3 NOUN+NUM 4 NOUN+FV}) }

NUM+CL {2 NUM+NOUN}

$\begin{bmatrix}\text{noun: } \alpha \\ \text{cat: X CL}' \text{ Y N}\end{bmatrix} \begin{bmatrix}\text{case: CL} \\ \text{sem: } \beta\end{bmatrix}$

acopy β ss.sem
delete CL′
copy_{ss}

NUM+NOUN {4 NOUN+FV}

$\begin{bmatrix}\text{noun: } \alpha \\ \text{cat: X N}' \text{ Y N}\end{bmatrix} \begin{bmatrix}\text{noun: } \beta \\ \text{cat: N}\end{bmatrix}$

ecopy β α
delete N′
copy_{ss}

NOUN+NUM {1 NUM+CL, 4 NOUN+FV}

$\begin{bmatrix}\text{noun: } \alpha \\ \text{cat: N} \\ \text{sem: } \beta\end{bmatrix} \begin{bmatrix}\text{cat: X N}' \text{ n}\end{bmatrix}$

ecopy α nw.noun
acopy β nw.sem
delete N′
copy_{nw}

NOUN+FV {}

$\begin{bmatrix}\text{noun: } \alpha \\ \text{cat: X NP} \\ \text{fnc:}\end{bmatrix} \begin{bmatrix}\text{verb: } \beta \\ \text{cat: NP}' \text{ V} \\ \text{arg:}\end{bmatrix}$

acopy α nw.arg
acopy β ss.fnc
delete NP′
copy_{ss} copy_{nw}

$\mathbf{ST}_F =_{def} \{(\begin{bmatrix}\text{cat: v}\end{bmatrix} \text{rp}_{\text{NOUN+FV}})\}$

It has been shown that the NP internal structure of Korean is somewhat more complicated than that of English due to some variation of word order and a rather free distribution of case markers inside the NP. Yet it has also been shown that those diverse constructions can be handled rather nicely in Database Semantics with little complication in the rule system as compared to that for English.

Bibliography

Bertossi, L., G. Katona, K.-D. Schewe, & B. Thalheim (eds.) (2003) *Semantics in Databases*. Berlin–New York: Springer-Verlag.

Bochenski, I. (1961) *A History of Formal Logic*, University of Notre Dame Press.

Chae, Wan (1996) "The meaning and function of Korean classifiers," *Chindanhakbo* 70. [in Korean]

Chang, Suk-Jin (1996) *Korean*, John Benjamins Publishing Company.

Copestake, A., D. Flickinger, C. Pollard, & I. Sag (2001) "Minimal Recursion Semantics: An Introduction," *Language & Computation*, Vol. 1 - No. 3:1–47, Hermes Science Publication.

Geach, P.T. (1969) "Quine's Syntactical Insights," in Davidson & Hintikka (eds.).

Hausser, R. (1999) *Foundations of Computational Linguistics*, 2nd ed. 2001, Berlin–New York: Springer-Verlag.

Hausser, R. (2006) *A Computational Model of Natural Language Communication*, Berlin–New York: Springer-Verlag.

Im, Hong-bin (2002) *The Depth of Korean Grammar III*, Taehak-sa. [in Korean]

Kamp, J.A.W. & U. Reyle (1993) *From Discourse to Logic*, Kluwer, Dordrecht.

Kang, Beom-mo (2002) "Categories and meanings of Korean floating quantifiers with some reference to Japanese," *Journal of East Asian Linguistics*, 11, Kluwer Academic Publishers.

Montague, R. (1974) *Formal Philosophy*, Yale U. Press, New Haven.

Minsky, M. (1974) "A Framework for Representing Knowledge," Techreport 306, Artificial Intelligence Laboratory, MIT. Republished in Brachman et al. (eds.) 1989 *Readings in Knowledge Representation*.

Miyagawa, Shigeru (1989) *Structure and Case Marking in Japanese*, Academic Press, New York.

Russell, B. (1905) "On Denoting" *Mind* 14:479–493.

Sohn, Ho-Min (1999) *The Korean language*, The Cambridge University Press.