LINEAR PHRASE STRUCTURE GRAMMAR

Myong Ro-keun, Park Hyo-Myong, Shin Gyonggu

The current model aims to give homomorphic explanation on syntax as well as semantics with linear operation, which is based on filled and to-be-filled categories. Filled categories are similar to slash categories of GPSG but with different effect. To-be-filled categories are provided as default value from ID rules. A sentence is not considered complete until to-be-filled categories are satisfied. The rules in this model are not only contex-free but also binary so that default value assignment becomes possible.

0. Introduction

In this paper we will try to construct a grammar model, which is similar to Generalized Phrase Structure Grammar (GPSG) in that it utilizes numerous conventions of GPSG, but is different in such important aspects as binary rules, linear-ordered rules, and homomorphism of syntactic and semantic operations. The current model, which is named as Linear Phrase Structure Grammar (LPSG), by no means claims to be a complete one. We only want the ideas introduced in this paper to contribute toward a more realistic grammar.[1]

1. Types of Grammar and Human Languages

Grammar is divided into four types based on the restriction levels. The

[1] More rigorous formulations and theoretical backgrounds of the current model are to be referred to the PH. D dissertation by Shin Gyonggu (1987). We want to appreciate Dr. Lee Hwanmook, who encouraged us to further polish the grammar model. He stands beside us as a godfather, while christening the model as a Linear Phrase Structure Grammar (LPSG). We also give deep thanks to Chang Kyonghee and Han Sungjin, who gave many helpful comments while scrutinizing this paper.

four classes are called Chomsky hierarchy, according to Noam Chomsky, who defined these classes as potential models of human languages. Type 0 does not have any restriction on the production rule as far as it satisfies this formula: $\alpha \rightarrow \beta$, where α and β are arbitrary strings of grammar symbols, with nonempty α. A string of length k might be derived from strings that are longer than k. In other words, it is allowed to generate a null string (Hopcroft and Ullman 1979: 220).

(1) a. Rule formula: $\alpha \rightarrow \beta$,
where α and β are arbitrary strings of grammar symbols,
with nonempty α.

b. Example grammar generating $\{a^i \mid i \text{ is a positive power of } 2\}$
$G_0=(Vn, Vt, P, S)$, where S is an initial symbol, and
$Vn = \{S, A, B, C, D, E\}$, $Vt = \{a\}$, and
P consists of

1) S → ACaB	5) aD → Da
2) Ca → aaC	6) AD → AC
3) CB → DB	7) aE → Ea
4) CB → E	8) AE → e

where e denotes a null string[2]

[2] What transformations achieved and failed: In their epoch-making paper, Peters and Ritchie (1973) proved that even the simplest fixed grammar with only two phrase structure rules can generate a recursively enumerable set. Generating a simple language with a complicated grammar is what we never want to get to. Furthermore, the transformational grammar G_t generates ungrammatical sentences, and it triggers continuous revisions of the grammar one after another. The following set-theoretical statement will make it clear what the overgeneration problem of TG is:

(i) $S=\{s_1, s_2, \ldots s_n\}$
where S is a set of grammatical sentences of English
$S'=\{s'_1, s'_2, \ldots s'_n\}$
$L(G_t)$=S union S'(or S unions with a subset of S')
where $L(G_t)$ is a set of sentences generated by a grammar G_t.

The following exposition will be of some help in reviewing what we have said about transformational grammar.

(ii) D=a set of sentences generated by context-free phrase PSG
E=a set of grammatical sentences of English
D is a proper subset of E.

What G_t aimed to do is to expand set D up to set E by transformational function T as (iii). But TG has gone too far ahead generating even ungrammatical sentences; the relation in (iii) does

Type 1 grammar is often called context sensitive (CS) grammar. It is in the form of $\alpha \rightarrow \beta$, where β is as long as α. Note that there is no such restriction on the right side of the rule in Type 0 grammar. In a context sensitive grammar, any of its production rules should have at least as many symbols on the right side as on the left. The term 'context-sensitive' comes from a rule formula of $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$, with nonempty β (Hopcroft and Ullman 1969: 12), where α_1 and α_2 indicates contexts.

(2) a. Rule formula: $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$, with nonempty β.
b. Example grammar generating $\{a^i b^i c^i \mid i > 0\}$
$G_1 = (Vn, Vt, P, S)$, where S is an initial symbol, and
$Vn = \{S, B, C\}$, $Vt = \{a, b, c\}$, and
P consists of

1) S → aSBC	5) bB → bb
2) S → aBC	6) bC → bc
3) CB → BC	7) cC → cc
4) aB → ab	

Type 2 grammar is called context free (CF) grammar. Production rules do not have context in the left, which means there is no more than one nonterminal string. The nonterminal symbol in the left side is replaced by string β in the right, independent of the context in which A appears. The contexts α_1 and α_2 in the rule (2), are simply null in context free rules.

(3) a. Rule formula: $A \rightarrow \alpha$, where α is not a null string

not hold, but E is a proper subset of T(D) as in (iv). So TG has tried in vain to restrict the set with the help of constraining function C on the power of T as (v); C(T(D)) is still a bigger set than E:

(iii) T(D)=E
(iv) E is a proper subset of T(D)
(v) C(T(D))=E

In addition to the problems of the power of the transformational grammar, there are some specific problems of individual rules such as nominalization (Chomsky, 1970), pronominalization (Bach-Peters Paradox by S. Peters 1973 and E. Bach 1970). All these problems have caused a series of drastic revisions of the grammar, finally arriving at a constraint-oriented grammar, Government-Binding Theory.

An important issue in the future seems to be how to improvise parsing procedures of the language generated by this grammar. As we mentioned in section one, a context sensitive language takes the time going up exponentially with the length of the input, but a language L_t generated by a transformational grammar would take time no less than a doubly exponential function of the sentence length (Gazdar 1982b: 133).

b. Example grammar generating $\{a^ib^i \mid i > 0\}$
$G_2=(Vn, Vt, P, S)$, where S is an initial symbol, and
$Vn=\{S\}$, $Vt=\{a, b\}$, and
P consists of; $S \rightarrow aSb$ $S \rightarrow ab$

Type 3 grammar is called regular grammar. A production rule not only includes no context in the left side as in context free grammar, but also should have terminal symbol optionally followed by a nonterminal symbol in the right side as in $A \rightarrow aB$ or $A \rightarrow a$. The nontermninal symbol in the left side is replaced by the string in the sequence of terminal and nonterminal strings indpendent of the context in which *A* appears.

(4) a. Rule formula: $A \rightarrow aB$, or $A \rightarrow a$
b. Example grammar generating $\{a^ib^j \mid i > 0, j > 0\}$
$G_3=(Vn, Vt, P, S)$, where S is an initial symbol, and
$Vn=\{S, A, B\}$, $Vt=\{a, b\}$, and
P consists of
1) $S \rightarrow aA$ 5) $S \rightarrow bB$
2) $A \rightarrow aA$ 6) $B \rightarrow bB$
3) $A \rightarrow bB$ 7) $B \rightarrow b$
4) $A \rightarrow a$

Because of the extreme simplicity of its production rules, the regular grammar is the least efficient in its generative power, while being the most efficient in parsing or recognition.

Symbolizing the grammar of type n with T_n, we can compare the generative power among the different grammars as follows, putting the more powerful grammar in front of the less powerful.

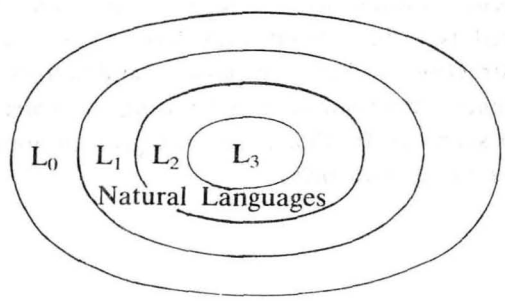
$$T_0 > T_1 > T_2 > T_3$$

And consequently, by defining L_n to be the language generated by T_n, we get the following subset relations;

$$L_0 \supset L_1 \supset L_2 \supset L_3$$

It is a widely accepted assumption that human languages are in a recursively enumerable (r.e.) set at most. If a language is a set of strings bigger than a r.e. (type 0) language, children can not possibly learn to speak or understand a natural language in a relatively short time in a given life span.

The next item we should take into account is the fact that speakers of a language quite quickly can tell the difference between grammatical sentences and ungrammatical sentences of their language. Thirdly, the mistakes that children make during acquisition of their native language clearly show that they are learning systematic rules as in *he goed* for *he went.* And we also assume that the grammar of human languages would not go below the power of the extremely restricted grammar of type 3. Chomsky's epoch-making book, *Syntactic Structures* (1957: 19), flatly denies the relevance of the *Markov* process to the human language, a transition network equivalent in its power to the regular grammar which was advocated by the structuralists. Our assumption on the scope of the natural language is, then, that it would be located somewhere between type zero and type three languages as:



Exclusion of type 0 and type 3 grammars from the list of natural language grammars has been common practice among formal linguists, while there has been much controversy on whether the CF grammar or the CS grammar would be more appropriate in describing the human languages. Common assumptions were that the CF phrase structure grammar (PSG) is not powerful enough to describe human languages, but both linguists and computer scientists have not stopped trying to find out ways to utilize the concept of context free grammar because of its efficiency in parsing. What is known on the time to be consumed for the context free language is:

(5) a. In the worst case
time = constant × length^3
b. In the best case
time = constant × length

Context-sensitive grammars have been considered the most promising candidates as grammar for human languages. Chomsky (1965) utilizes this

type of grammar extensively in the subcategorization schema of the verbs. Its relative inefficiency in parsing, however, has been a major reason of reluctance to accept it as a feasible grammar (Bach 1973: 195). Augmented Transition Network, or ATN, which commonly deals with the context sensitive grammar, takes the time going up exponentially with the length of the input (Charniak and McDermott 1984:221).

(6) time = constant $\times$ 2^{length}

Though the type 2 grammar has a great advantage in its parsing time over type 0 and type 1 grammars, a traditional context free phrase structure grammar has been considered not to be powerful enough to generate all the sentences of a natural language. Based on monadic category node, it loses cross-categorial generalizations as between verbs and adjectives even though both are considered as verbals. It does not have cross-structural generalization in some constructions as between passives and actives, between normal and inverted sentences, between equi and raising constructions. One crucial drawback of PSG seems to be that it can not give an appropriate explanation on discontinuous dependencies.

2. GPSG

For a grammar model to be a plausible candidate for a natural language, we assume that it should explain the problems we talked about in the previous section. In addition we require it to be context free to create an efficient parsing model as well as syntactic generation and semantic translation.

2.1 Mono-Stratal Syntax

Starting at the wrong place is quite costly even with a correct orientation. That is how the transformational grammar seems to have been. It takes almost two decades to get rid of the transformational rules, and to go back to the surface structure. The surface sentences are what the layman speaks, hears and thinks with. Even the trained linguists use the surface constructions when they are out in the street. While indulged in the complicated theorization, the generativism has been far away from this ordinary people's intuition. But on the other hand, the orientation set by Chomsky (1965) has

contributed to the syntax by setting a correct direction: if a theory of grammar is explanatorily adequate, it should be psychologically realistic. The ruler of psychological reality finally has taken some of the syntacticians back to the starting point of mono-stratal perception of the languages. Dual--stratal explanation of sentences forces a theory to be too complicated to be psychologically realistic (Fordor 1983). Matching the surface structure with the deep structure is far more time-consuming than necessary.

2.2 Context Free Grammar

The current version of LPSG is an offspring of G. Gazdar, E. Klein, G. Pullman and I. Sag (1985 henceforth GKPS), and these two grammars share many features in common. One of the most important features they share is context-freeness. Gazdar (1982) points out two main reasons for keeping the context-freeness: (1) Natural languages are learnable, while the class of languages by a transformational grammar is unlearnable since they belong to the recursively enumerable set; (2) Parsing of a natural language is quick and easy and sentences generated by a context-free language are probably parsable in the time which is, at worst, proportional to less than the cube of the sentence length or mostly in a linear time. One of the assumptions of a psychologically realistic grammar is that the grammar should be simple enough to mirror the easiness of the learning process and the quickness of uttering and recognizing sentences by pseakers and hearers. As we have already seen, the most promising candidate satisfying such basic assumptions is the contex-free grammar. But many of the linguistic phenomena have been considered to be inexplicable through context free rules. The fact that the same category, say, a verb category functions in different ways from others as in (7) has been disappointing to those who want to render context--freeness to the natural language. Accordingly most of the phrase structure rules of English are devised as context-sensitive rules as we see below (Bach 1973, GKPS 1985:48).

(7) V → V/__ NP to NP
V/__ NP for NP
. . .
V/__ about NP
V/__NP
V

Different context-free rules are contrived with subcategorized rule num-

ber in (8) for each of the above context-sensitive rules in (7). The set of context-free rules below are exactly equivalent in its generative power to the set of context-sensitive rules in (7). GKPS shows us additional advantages of the context-free phrase structure rules with subcategorization such as in coordination (GKPS 1985:35).

(8) VP → V1 NP to NP
VP → V2 NP for NP
. . .
VP → V10 about NP
VP → V11 NP
VP → V12

A further strategy of changing a context-sensitive grammar into a context--free grammar is shown below with the subject-verb agreement which is traditionally considered to be treated by a context sensitive rule (M. Gross 1972:132). Rule (9) is converted into two context-free rules in (10):

(9) VP → V[+Plu] / NP[+Plu] ____

(10) S[+Plu] → NP[+Plu] VP[+Plu]
VP[+Plu] → V[+Plu] NP

But this feat is not accepted by some grammarians on the ground that we will need a much expanded set of rules. But the expansion of rules does not seem to cost much. As we are looking up a lexical item from the lexicon, we can simply pick up a rule from a grammar and use it. GKPS, however, accepted this criticism and gives a solution by replacing rules as in (10) with control-agreement principle (CAP), which will not be discussed in detail.

2.3 Simple Enumeration of Rules are More Efficient

The following are cases where generalized simplification of (12) is better than simple enumeration of rules of (11). And input-output relations of (11) may well be revised into one simpler rule of (12). Rule (12) is not merely a generalization but possibly a mirror image to the way the German native speaker thinks. (# indicates a word boundary, and $ a syllable boundary)

(11) b → p/__ {C, #}
d → t /__ {C, #}
g → k/__ {C, #}

s → z / _ {C, #}

(12) [−Delayed Release, −Strident, +Voiced] → [−Vioced] / _ $

On the contrary, in some cases, looking-up more rules is less complicated and costly in terms of the parsing time than deriving rules from some general rule-producing schemata such as passive metarules. Verb inflections in (13) do not get a generalized simplification. But it is easy to see that looking-up through the list of inflected words will cost less time than finding *see* first then picking up *saw* or *seen.*

(13) Enumeration of lexical items:

. . .
saw [see′, PAST]
see [see′, BASE]
seen [see′, PAST PART.]
. . .
walk [walk′, BASE]
walked [walk′, PAST]
walked [walk′, PAST PART.]
. . .

As many of the adult language learners of a foreign language might have experienced, it is hard, or rather impossible to look up an inflected form of a word in some dictionaries which do not enumerate inflected forms but include the base forms. We can get the idea even from a familiar language. For example, when we express the idea 'we *saw* a unicorn,' we do not look up *see* first before going to *saw* to say 'I *saw* a unicorn.' Note also the fact that it is a common practice to enumerate all the products of derivation and inflection rather than to include them under the base forms in the computerized dictionary. It is because the former way of enumeration is faster in looking up an inflected form of a word. It will be time-consuming to look up *seen* in the following dictionary, even though the list itself is longer with dictionary (13) than with dictionary (14).

(14) Rule-governed dictionary:

. . .
see [see′, BASE]
[see′, PAST] → saw
[see′, PAST PARTICIPLE] → seen

. . .
walk [walk′, BASE]
[walk′, PAST] → walked
[walk′, PAST PARTICIPLE] → walked
. . .

We will not follow the conventional way to simplify the number of rules to simplify the grammar to the extent that only a few different rule schemata, for example, are allowed (Pollard 1984: 74f, Pollard and Sag 1987: 13, 147ff). However, the category unification principle or principles of universal (English) grammar (Pollard 1984: 50f, Pollard and Sag 1987: 145f) is complicating the process of introducing syntactic structures more than necessary. For example, Pollard (1985: 51, Pollard and Sag 1987: 13) sets up bare PS rule schemata to be unified to generate new structural description.

(15) Scheme:
a. [SUBCAT < >] → H[LEX −], C
b. [SUBCAT < >] → H[LEX +], C*

(16) a. Phrase structure rules instantiated from (15a):
S → NP, VP
NP → DET, NOM
NP → NP's, NOM
b. Phrase structure rules instantiated from (15b):
VP → V; VP → V, S′; AP → A;
VP → V, NP; AP → A, PP; PP → P, NP
VP → V, PP; VP → V, VP; VP → V, AP
VP → V, NP, NP; VP → V, NP, PP; etc.

With the above rule schemata in (15), the only way to complete the tree is to instantiate the H first from one of the relevant lexical entries. And only afterwards it is possible for C to be instantiated. But in recognition process there is no way for the parser to begin from the left node, which is normally not a head constituent in SOV and SVO languages. The farther the head is located from the first node, the less efficient becomes the parsing process. To see the acceptability of a given sentence it should go to the head and come back to the beginning to start to examine whether each complement is given legality by the head.[3]

[3] In the following example, a parser of head-driven grammar will waste time until he/she hit

2.4 Feature-Based Categories, and Syntactic Operations

GPSG extensively utilizes syntactic features to the extent that it asserts that a categorial node is not monadic but a set of features. It does not only help capture the cross-categorial generalization, but its feature instantiation principles such as head feature convention (HFC) and foot feature principle (FFP) enriches the set of features inherited by the immediate dominance (ID) rules with functional informations, cases, tenses and even informations on displaced nodes and equi nodes. These features are classified into two: **head** and **foot.** In this paper, we will exclude *slash* from foot, and instead postulate **default** features as in (17c). They are different from both head and foot features. They are operational features instantiated by default feature assignment principle, which will be explained later in detail.

(17) a. HEAD = {N, V, PLURAL, PERSON, VERBFORM, SUBJ, PREPFORM, AUX, ... SLASH, AGREEMENT, SUBCAT, BAR-LEVEL, LOCATIVE}
b. FOOT= {WH, REFLEXIVE/RECIPROCAL}
c. DEFAULT= {FILLED, LACKED}

Concept of category has been expanded with syntactic features; a different set of features results in a different category. When two different categories, or two different sets of features, share a certain number of features, they are operationally related in respect to those shared features.

2.5 ID Rules and LP Statements

GPSG decomposes one PSR into two distinct relations: immediate dominance (ID) and linear precedence (LP). But GPSG seems to be the first enterprise to utilize this concept so extensively, expanding expressive power

the head *ate*. With the length of the subject becoming longer, waste of time will increase.

(i) a. Jack *ate* a frog.
b. The boy *ate* a frog.
c. The young boy who did not *eat* anything for a week ate a frog.
e. The young boy who ... *ate* a frog.

Head-driven grammar requires bottom-up parsing, since there is no mother node given in advance. A further trouble is that it is extremely inefficient to accept the node corresponding to an *empty node*. It can be done only after we find an *empty node* in the i-th position and go up to the mother and down to the node corresponding to the trace.

of the PSRs by the ID relation factorized away as in (18). Rule (18) only states that VP is composed of V and NP in the given order. But with the LP statement absent, the expressiveness of the rule is expanded as from (18) to two PS rules of (19b).

(18) VP→ V NP
(19) a. VP → V, NP
b. VP → NP V
VP → V NP

The simplifying effect is significant in the case of free word order languages such as Korean and Japanese. With the concept of normal PSR we need to have two different rules for transitive verbs, but these two rules in (20a) are reduced into one as in (20b) with LP statement factorized out as in (20c).

(20) a. S → NP[NOM] NP[ACC] V
S → NP[ACC] NP[NOM] V
b. S → NP[ACC], NP[NOM], V
c. NP < V

We can find the effect of the current revision more easily with more nodes added before V. In the case of the following rule (21a), we are given 38 PSRs which are reduced into one ID rule with another LP statement (21b) following. Rules (21a) and (21b) are more general and informing than those which might have been expressed by a number of normal PSRs in (21c).

(21) a. S → NP[NOM], NP[ACC], (PP), (ADVP), V
b. α < V, where α is a set of NP[NOM], NP[ACC], PP and ADVP
c. S → NP[NOM] NP[ACC] (PP) (ADVP) V
S → NP[ACC] NP[NOM] (PP) (ADVP) V
S → NP[NOM] NP[ACC] (ADVP) (PP) V
. . .

LP statement on the other hand is restrictive in its expressive power, but it also gives an important generalization on English word order, suggesting a cross-rule generalization as in (22): a lexical item of the feature of [+ SUBCAT] always precedes its phrasal sister with the feature of [−SUB-CAT]. Staying in the same line, we may well simplify the LP statement on Korean of (21b) into (21b′).

(22) [+SUBCAT] < [−SUBCAT]
(21) b′ [+SUBCAT] > [−SUBCAT]

All these mechanisms and most of the conventions will be assumed to be operating in our grammar except matarules such as passive metarule, slash termination metarule and subject-aux inversion metarule. The slash termination metarule is going to be unnecessary by ID rule default. The passive metarule will be simply disregarded, since we consider the passive-active relation is explained by lexical redundancy rule and semantic translation. We simply do not postulate the existence of subject-aux inversion, and it is semantically related with uninverted construction.

3. Linear Phrase Structure Grammar (LPSG)

3.1 General

One of the main objectives of this paper is to limit the grammar generating English sentences within the realm of CF grammar with both the recognition and the acquisition process in mind. The system we adopt here, however, goes beyond that; we will constrain further the CF grammar of GKPS to be a binary grammar, maintaining its expressive power equivalent to that of CF rules of GKPS (1985).

The reason we adopt a binary CF grammar is based on the psychological reality; we assume that a given sentence is processed two by two. In addition to the binary context-freeness of the grammar we have an additional requirement from psychological reality: linearity. This requirement derives from the consideration of the recognition process. Studies on the recognition process have already contributed to changing grammarians' attitudes toward the phrase structure rules, which is no longer considered as a generative process but as a node admissibility or well-formedness condition. We will go one step further and take the rules to be linear-bound from left to right simply because we commonly do so while processing sentences. The linearity of the grammar gives rise to a further revision on the way of semantic translation; the translation is to follow the linear order of a sentence. And in contriving the way of logical translation, the prime concern is to be on the linear order of a sensence but not of the sequence of composition from right to left in a traditional way.

The syntactic revolution of GPSG mainly depends on expanding the ex-

pressiveness of the category by considering it to be a set of flexible number of features, whether inherited or instantiated. A further revision of the concept of cateogry is to meet the requirements caused by the binary grammar and linearity. We will expand the concept of slash category. A [SLASH] category of GPSG is considered as a category already **filled.** The opposite of the filled category is a **lacked** category still to be filled to make the sentence syntactically or semantically complete.

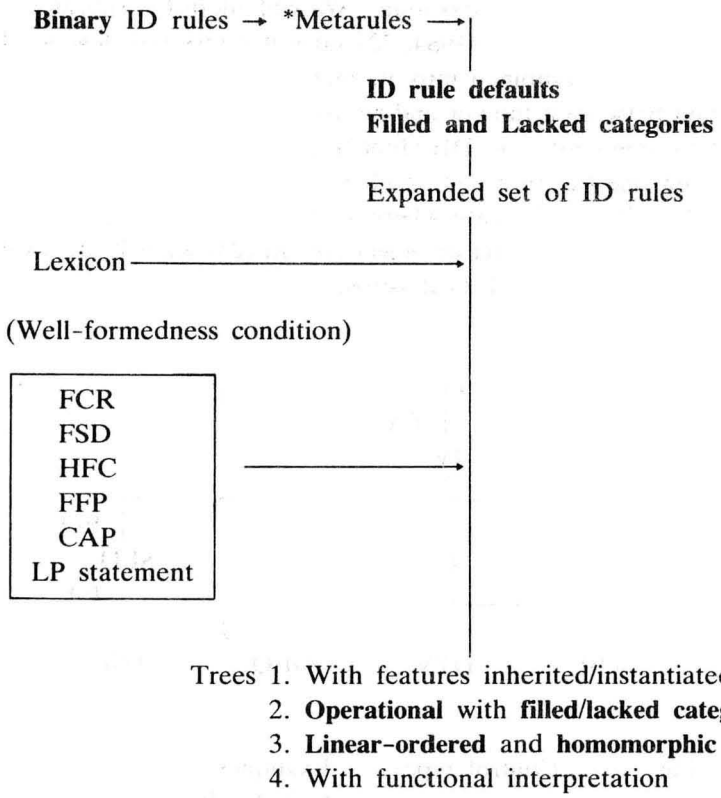
For the lexical entries we will utilize some ideas of Pollard (1984, 1985a, 1985b). The three bar-level will be accepted only for NP but not for VP. A verb to be combined with NP only once is a different category from other verbs to be combined with NP twice. This assertion amounts to complete denial of the bar-level.

One of the important means of theroretical explanation in GPSG is a metarule mechanism. But we will disregard it because of its over-generation problem cited in Peters and Uszkoreit (1982, double quoted from Pollard 1984: 6). Lexical redundancy rule and word formation rules will take care of the task of explaining cross-categorial generalization. Cross-structural generalization will be obtained by shared syntactic features as well as semantic translations.

There are some important revisions in this paper. One of the most significant ones is **default values** in the sentential level, which are extensively used to explain from simple sentence parsing to discontinuous dependencies. They derive from ID rules, both lexical and phrasal. Since these features are given in the local node, they percolate down or up between a dominating node and dominated nodes. The trickling down/up is made possible by feature instantiation schemata such as FFP and HFC. The following diagram which was modified from Sells (1985) shows how the LPSG is similar to and different from the model of GKPS.

The major differences of this LPSG from GPSG is that LPSG adopts the strictly linear grammar in which syntactic and semantic operations are homomorphic. The linearity of grammar is a consequence of extensive use of ID rule defaults, which is in turn made possible by constraining the grammar within the realm of binary grammar.

LPSG (Bold-faced items are major differences of LPSG from GKPS, and star-marked items are no longer operating in LPSG)



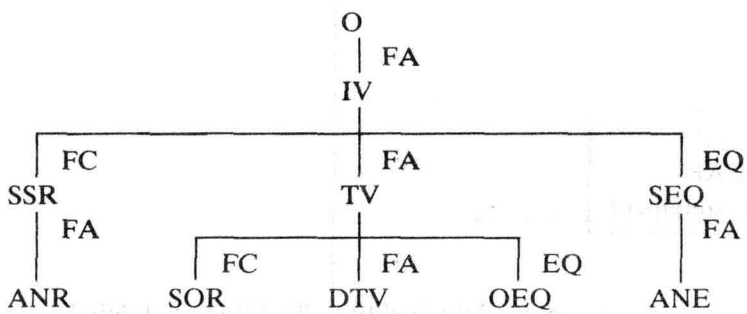
3.2 Category Level and Bar Level

In order to make possible a linear-odered binary grammar with homomorphic operation of syntax and semantics, we adopt the category level and semantic categorization which are similar to those of Pollard (1984), which again follows the theoretical construct of the categorial grammar. Consequently, X-bar will not be considered. The disposal of the phrasal bar level is required by the binary combination of constitutents. And we consider that X-bar syntax is not much of a generalization in the lexical verb categories.

For example, *kill* and *give*, which have been considered to belong to the

same bar-level, are different from each other in their way of combination. The former is combined with an object NP to be a VP, while the latter is combined with an object NP and again with another object NP to be a VP. *Love* has a different compositionality from *give*. The different category levels decide its future career of composition. We will adopt the subcategorization schema of verbs by Pollard (1984: 32),since it shows how low-level subcategories of verbs are combined into another. **FA** represents a normal fuctional application between a functor and an argument, in which a functor is combined with an argument as in DIE(John′). **FC** represents a functional composition between one function and another to become a function as in (seem′(go′))(x). **EQ** indicates the case where an argument for a constituent VP is provided by the matrix subject or object NP. In (23) we will see a list of different subcategory types of lexical verbs.

(23)



(24)

Name	Description	Control type	Examples
O	no arguments	Null	S, NP, PP
IV	intransitive	⟨FA⟩	intr. verb, VP, predicative
TV	transitive	⟨FA, FA⟩	TV, TV Phrase, preposition
SSR	subject-to subject raising	⟨FC, FA⟩	*tend, certain,* AUX
SEQ	subject equi	⟨EQ, FA⟩	*try, eager*
DTV	ditransitive	⟨FA, FA, FA⟩	*give*
SOR	subject-to object raising	⟨FC, FA, FA⟩	*believe*
OEQ	object equi	⟨EQ, FA, FA⟩	*persuade*
ANR	anomalous raising	⟨FA, FC, FA⟩	*seem*
ANE	anomalous equi	⟨FA, EQ, FA⟩	*promise*

More examples on the semantic translation of the different types in a sentential context are given later adopted into our linear grammar. NP's are also classified into different subcategorizations as in the case of verbs. The difference of NP from VP is that it has only three levels of categories: lexical, semi-phrasal (NOM) and phrasal.

3.3 Two Kinds of Default Categories

The linear-ordered grammar from left to right is by no means newly-fashioned. It is all too common in the parsing strategies in the computer science. It is intuitively plausible as well. No speaker seems to generate or recognize sentences in the order from right to left. Furthermore, no one seems to generate sentences in one way (say, from left-to-right) and recognize them in the other way (say, from right-to-left-to-right). We will assume that a psychologically realistic grammar should be in linear-ordered operation. A hearer often guesses what kind of constituent or word is coming next, while the other side is speaking. He frequently picks up a word or a phrase with a correct syntactic category, while the speaker is at a loss what word or phrase to select. This kind of fact suggests to us that a human being is capable of parsing sentences in a linear order from left to right.

In order to make possible a linear-ordered operation we will extend the convention of slash categories, which is proposed in GPSG, originally in J. Bear (1981), to **default category.** The conventional slash category of GPSG is similar to the **filled category** of LPSG in that it is already given and under feature instantiation convention. It is also different from the filled category of LPSG in that it is a filler category of the following gap. Since there is no gap, the filled category of LPSG is not to be repeated in the sentence. If it is repeated it is unacceptable as follows:

(25) a. *John,* Mary loves.
b. **John,* Mary loves *John.*

The **lacked cateogry** is a category to be filled. For example, once we get a subject NP dominated by an S, a VP is obligatorily required. And at the node of subject NP as in (26), we can say that a VP node is a lacked category. When we are on the node of IV dominated by VP as in (27), no node is required or lacked. When we are on the node of *be* dominated by VP as in (28), ADJ node is required. In most cases such a lacked category is easy to find, since it is provided by relevant ID rules. And it is not even necessary to specify which is lacked or not, since we simply know it as a

default value. We will call it **ID rule default,** since this category feature is provided by ID rules. This is not a usual kind of slash category in terms of the filler-gap relations. The slash category of GPSG is a filled category in our LPSG. While the slash categories are commonly introduced by derived rules in GPSG, our filled/lacked cateogries are directly introduced by ID rules.

For semantic interpretations, we will use bold-faced, **P, Q,** and **R** as NP type variables, and **V** as a VP variable. Lambda abstraction is done on the lacked categories. If an abstracted argument or a function is not quantified in, the string is semantically ill-formed or incomplete. The quantification is done on the principle of first-come-first-served. And the semantic operation is carried out from the left to the right.

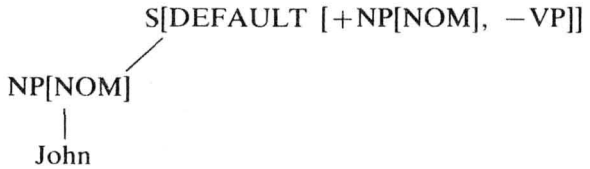
(26) S → NP[subj] VP: VP′(NP)

(27) a. VP → IV: λ **P**[V(**P**)]
b. die, eat, run, ...
c. runs: λ **P**[run′(**P**)]

(28) a. VP → be: ADJ; λ V λ **P**[V(**P**)] (*be* is a category of VP[−AP])
b. be
c. is; λ **P**[λ V[V(**P**)]]
is sick: λ **P**[λV[V(**P**)]](sick′)= λ **P**[sick′(**P**)]
John is: λ **P**[λ V[V(**P**)]](John′)= λ V[V(John′)]

In the example below, the process of generation and recognition begins with NP[NOM], which provides S with filled and lacked features. Unless a node is given, no information on a sentence is available, whether syntactic or semantic, and no operation is initiated. S node is assigned [DEFAULT {+NP[NOM], −VP}]. NP[NOM] is no longer expected to come, and [VP] is still lacked.

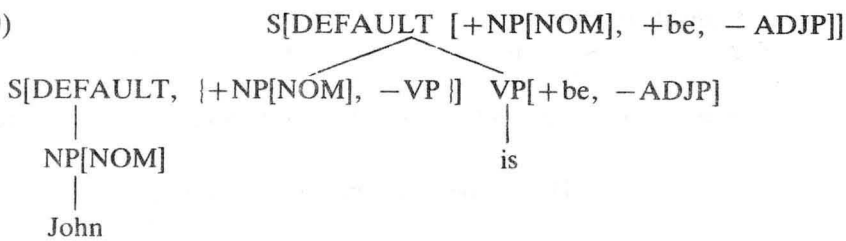
(29)



With a new lexical item *is* provided, the VP is partially completed but lacking AP, and we have a tree of (30). In the following case, *be* is considered a category requiring an adjective pharse. In the local level, the lacked category is provided by ID rule default based on the ID rule of *VP → be,*

AP. Leaving the local level, it is specified in the dominating VP node.

(30)



(From now on, the category-valued feature of DEFAULT will not be represented in the tree. Filled categories will not be specified unless a certain context require that it be specified.)

The left sister in most rules can be considered as a category requiring its right sister, and vice versa. For example, *be* is considered as a category lacking ADJP, PP[place], or NP complement, and all the following three categories require *be* to precede them. An intransitve verb is a category which does not require any NP[ACC] but NP[NOM], while a transitive verb is a category requiring both NP[NOM] and NP[ACC], and so it goes on. Semantic version of lacked feature is homomorphic to the syntactic version: If an NP translation is provided it is not a complete sentential translation until it is given a VP translation.

3.4 Case Marking and ID Rule Default

Our assumption is that a syntactic category, if sufficiently subcategorized, is able to show us what category is coming after it. In the case of NP, case marking is crucial in our grammar since we assume that the grammatical relations are not decided by the syntactic context as commonly considered (Chomsky 1965: 68ff, 93f). On the contrary, the case marked on an NP decides the syntactic context and syntactic order. In that sense, our model is a case sensitive grammar, but not a context sensitive grammar. For example, given NP[NOM] we expect a verb phrase, lexical or phrasal, to follow after it. With NP[ACC] provided, we expect to have TV as well as NP[NOM]. With a simple NP there is no way to predict what category will follow. This is why we are going to put onto NP nodes informations on case. NP[NOM] is theoretically equivalent to S[−VP], an S node lacking VP, and it needs to be combined with a VP. All the diagrams in (32) are equivalent to each other, and may well be generalized as (33).

(31) ID: S → NP[NOM], VP
LP: NP[NOM] < VP

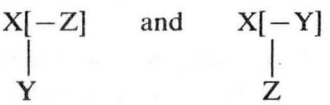
(32) Equivalent trees:

a. 

b. 

c. 

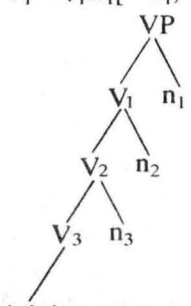
(33) Principles of ID rule default category assignment:
If there is a binary rule of the form X → Y, Z
then Y is equivalent to X[−Z], and
Z is equivalent to X[−Y],
or



All three trees in (32) are theoretically possible, but we commonly do not use (32b) nor (32c) since all the information in (32b) and (32c) is given as default values by rule (31) and generalization (33). However, we will use such information as in (32b) and (32c), whenever necessary, for linear-ordered operation. Limiting our concern on verbs, default value assignment principle can be schematized as in (34): (34a) for left-branching tree and (34b) for right-branching tree.

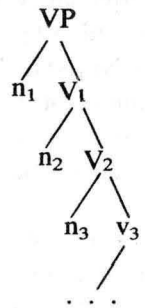
(34) a. Default value assignment for left-branching:

$V_1 = VP[-n_1]$
$V_2 = V_1[-n_2] = VP[-n_1, -n_2]$
$V_3 = V_2[-n_3] = VP[-n_1, -n_2, -n_3]$
. . .
$V_i = V_{i-1}[-n_i, -n_{i-1}, \ldots -n_1]$



b. Default value assignment for right-branching:

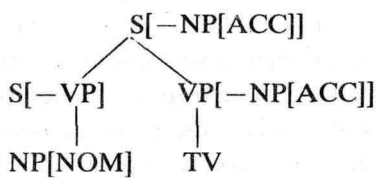
$n_1 = VP[-V_1]$
$n_2 = V_1[-V_2]$
$n_3 = V_2[-V_3]$
. . .
$n_i = V_{i-1}[-V_i]$



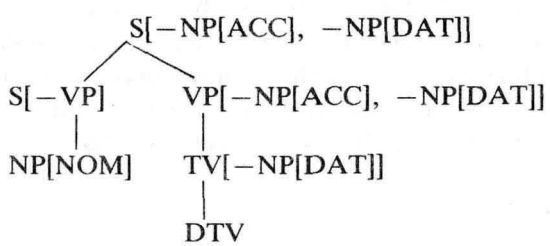
Theoretically such an extraction goes on infinitely, but in English it is limited to a maximum of three, including subject NP extraction as in the case of ditransitive verb, *give*. Each of X, Y and Z is a slash category automatically given by ID rule default and to be filled later for VP to be complete. For example from the set of rules in (35), ID-rule default will give us trees in (36), provided that we have only NP[NOM] and a verb:

(35) a. S → NP[NOM], VP
b. VP → IV
c. IV → TV, NP[ACC]
d. TV → DTV, NP[DAT]

(36) a. S → NP[NOM], VP[−NP[ACC]] for the tree of:



b. S → NP[NOM], VP[−NP[ACC], −NP[DAT]] for the tree of:



We consider VP, TV or DTV as mere mnemonics. TV and VP[−NP[ACC]], in fact, are equivalent to S[−NP[ACC], −NP[NOM]]; and DTV to TV[−NP[ACC]] and VP[−NP[ACC], −NP[DAT]], or to S[−NP[ACC], −NP[DAT], −NP[NOM]]. Some of the readers might worry about expansion of the number of complex feature categories of the set X, Y and Z. But it is rule-generated, and thus we do not need to memorize all of the members. The language learner simply memorizes basic ID rules as in (35a) to (35d).

This kind of convention on ID rule default is only possible by constraining the CF grammar into binary grammar, so it seems to be an appropriate time for us to deal with the binary grammar in the next section.

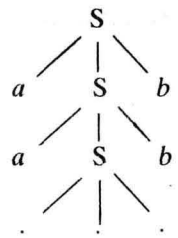
3.5 Binary Grammar

In order to utilize the default category which we discussed in the previous section, we add a requirement for the linear-ordered grammar: the grammar should be binary. In other words, the ID rules can not bear more than two children; the third child is illegal. This binary property is also crucial in explaining the homomorphism between syntax and semantics, which implies that syntactic operation as well as semantic translation should be binary. First we will show that the binary grammar and the CF grammar can be equivalent in its expressive power.

There has been much effort to restrict the grammar in the realm of CF grammar, but no effort to confine it with the binary composition. Furthermore, incompatibility of syntactic and semantic operation is simply disregarded. It has been considered natural to carry out syntactic operation in one way and semantic operation in another. Even semantics-oriented GPSG constructs a semantic operation independent from syntactic operation. Following categorial semantics, we will consider the semantic operation should be binary. And then in order to get homomorphic operation between syntax and semantics, we should show that the power of the binary branching grammar can be equivalent to that of the normal CF grammar. Once we achieve this goal the grammar will be nearly as efficient as the regular grammar which is a binary branching. One of the most common examples of CF grammar is the following one which was introduced in the first section of this paper, which shows a string with a recursively infinite number of embedded strings of equal number of *a*'s and b's: $a^n b^n$.

(37) a. S → aSb
S → ab

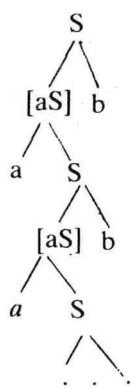
b.



We can revise a binary branching grammar without reducing the power of a normal CF phrase structure grammar with expanded function of nodes. The strings generated by the rule given above can also be generated by the following rule only with binary branching by introducing a new node of [aS] representing the sequence of *a* and *S*. The only difference between the former and the latter grammar is that the latter can have one more string of *a*, which the former does not have. With one rule of 'S → a' added to the binary grammar, the two become completely equivalent.

(38) a. S → [aS] b
[aS] → a S
S → ab

b.



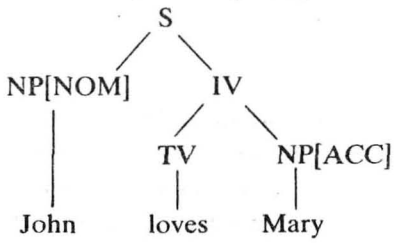
The revised version generates a string of equal number of *a*'s and *b*'s. This is an important step toward having a syntactic operation with homomorphic semantic translation, for the semantic translations are carried out binarily.

However, there is a further obstacle in achieving a linear grammar: we need to map ID rules into a tree which shows a homomorphic operation from left to right. Traditionally the semantic translation is done from right to left with bottom-up operation, which exactly corresponds to the tradi-

tional tree diagram in (40). (In the diagram below, the node IV is considered to be equivalent with VP, since both are composed with an NP[NOM] to be an S.)

(39) S → NP[NOM], IV
IV → TV, NP[ACC]

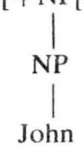
(40)



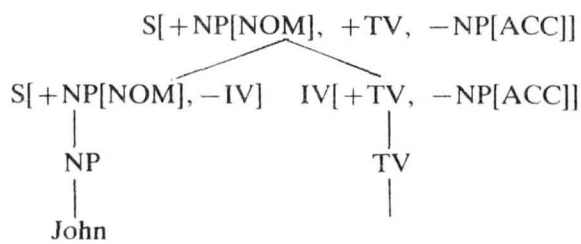
This tree does not explain left-to-right linear-ordered operation. So we will revise the tree diagram which will allow node combination from left to right, using ID rule defaults. The tree will be made through left branching operation instead of right branching for the sake of linearity. Consequently, the above tree will be revised as in (42). And we will assume the tree to be not a final product, but a process from the left-most node to the right-most node.

(41) *ID rule defaults assigned by principle (33):*
S[+NP[NOM], −IV] from the rule S → NP[NOM] IV
IV[+TV, −NP[ACC]] from the rule IV → TV NP[ACC]

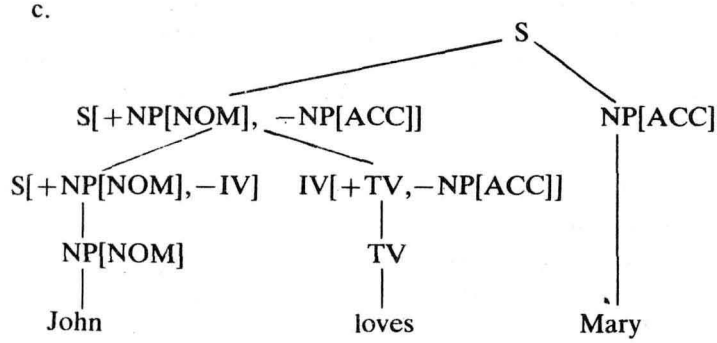
(42) a.



b.



c.



The fact that a category of a node resulting from composition is not a legal one, gives rise to no problem. For in our grammar, a category is simply defined as a set of features instead of a rigid unit. More concrete examples will be given later for more semantic translation.

3.6 ID Rule Defaults and Getting Rid of Metarules

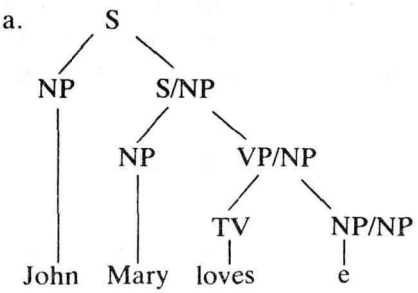
One of the by-products of the current use of ID rule defaults is that we do not need the slash termination metarule (STM). If our grammar does not generate a trace in the tree, there is no need of slash termination metarule to get rid of the trace. The following example is generated from (44). With a traditional way of tree diagramming, we will have a trace as in (45).

(43) *John,* Mary loves

(44) a. S → NP VP

b. VP → TV NP

(45) a.



But with the ID rule default, no empty node occurs. A filled category simply is not expected in any way. If an unexpected node appears, the sentence overflows with identical nodes and is judged to be unacceptable. With the following example, we will see why there is no trace introduced in the tree. Filled categories are redundantly specified to show how the ID rule

default operates. During the tree building operation we assume convention (47) is working. Filled categories are commonly not represented in the tree. In the following tree we will not give attention to the sequence of NP[ACC] followed by NP[NOM], since it is considered to be decided upon by the LP statements.

(46) ID rule default category assignment by principle (33):
  a. S lacking VP:
     S[−VP] → NP[NOM] from S → NP[NOM] VP
  b. S lacking NP[NOM]:
     S[−NP[NOM]] → VP from S → NP[NOM] VP
  c. VP lacking NP[ACC]:
     VP[−NP[ACC]] → TV from VP →TV NP[ACC]
  d. VP lacking TV:
     VP[−TV] → NP[ACC] from VP →TV NP[ACC]

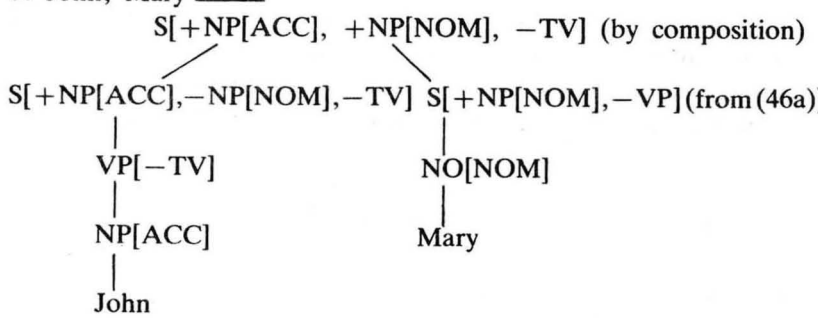
(47) Composition Convention:
  a. Filled and Lacked features of a node are given by ID rule default assignment principle (33). (Examples in (46))
  b. A lacked feature [−X] is converted into a filled feature [+X], when it hits X node. (Examples in (48b))
  c. A lacked feature [−X] is replaced by the default categories of its daughters, [+Y] and [−Z], when it hits one of them. (42b)
  d. A lacked feature [−X] is erased by its single daughter node [+Y], when it hits it. (48c)
  e. The highest S node is not replaced by its daughter nodes.

The concept of tree is radically different from the conventional ones. It is considered as a process of syntactic and semantic operation. It is also a process of generation or recognition. Below we will only think about a syntactic operation.

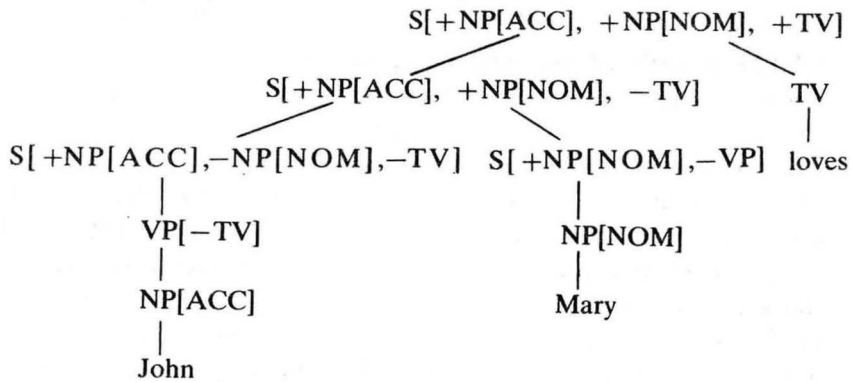
(48) Tree diagram of *John Mary loves* based on the ID rule defaults:

  a. John_____
     S[+NP[ACC], −NP[NOM], −TV] (from (46b) and (46d))
     |
     VP[−TV] (from (46d))
     |
     NP[ACC]
     |
     John

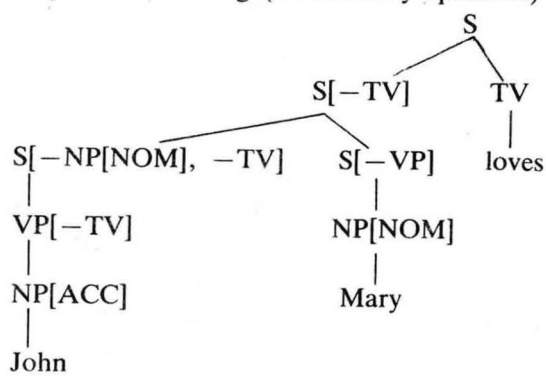
b. John, Mary ——



c. John, Mary loves.



c′. Simplified excluding (redundantly specified) filled categories:



The S node is lacking nothing until it is given an initial node. After having been provided with a node, ID rule default is activated to introduce a lacked category, which should be satisfied somewhere in the tree keeping well-formedness condition such as LP statement.

As we see in the above operation there is no mechanism whatsoever for trace or slash categories to be introduced. Introducing an empty node is equivalent to saying that there is a rule for generating a null string. If we want to keep our grammar within the boundary of context-freeness, it is required that we avoid rules generating a null string. Note that there is no context free rule which is capable of generating a null string. Only Type 0 grammar is able to have a rule generating a null string. (Refer to section one for the difference of power of different types of grammar). It is also worth while to note Pollard (1984) who asserts that the language generated by such a grammar amounts to a language generated by an unnecessarily powerful Type 0 grammar.

3.7 Linear-Ordered Semantic Interpretation

Our assumptions are that syntactic and semantic operations are homomorphic and that they are linear ordered. Consequenlty it is necessary for our semantic interpretation to be linear-ordered. It should be carried out from left to right in the order of words of a sentence. Any element coming first will be translated first into the semantic representation. Thus in SOV language it is translated as in (49a), subject first and then object later. In OSV language it is translated as in (49b), object first and then subject later. The current semantic operation implies that any node coming first should be considered as a functor while the node immediately following it should be considered as an argument. The following examples are prepared to illustrate that sentences in any sequence of any language are translatable in their given order.[4]

(49)

a. SOV:

John-Subj:	λ V[V (John′)]
John-Subj Mary-Obj:	λ V[V(John′)](λ **P**[λ V[V(Mary′)(**P**)]] $\Rightarrow$ λ **P**[λ V[V(Mary′)(**P**)]](John′) $\Rightarrow$ λ V[V(Mary′)(John′)]
John-Subj Mary-Obj loves	λ V[V(Mary′)(John′)](love) $\Rightarrow$ love′(Mary′)(John′)

b. SVO:

John-Subj:	λ V[V [John′)]

[4] The linear semantic operation of different word order (Shin 1987) is revised into a current form, following the comment of Cho, Kyongsuk.

John-Subj loves: λ V[V(John′)](love′) $\Rightarrow$
λ V[V(John′)](λ **P**[λ **Q**[love′(**Q**)(**P**)]]) $\Rightarrow$
λ **P**[λ **Q**[love′(**Q**)(P)]](John′) $\Rightarrow$
λ **Q**[love′(**Q**)(John′)]

John-Subj loves Mary-Obj: λ **Q**[love′(**Q**)(John′)](Mary′) $\Rightarrow$
love′(Mary′)(John′)

c. OSV:

Mary-Obj: λ **P**[λ V[V(Mary′)(**P**)]]

John-Subj: λ **P**[λ V(Mary′)(**P**)]] $\Rightarrow$
λ **P**[λ V[V(Mary′)(**P**)]](John′) $\Rightarrow$
λ V[V(Mary′)(John′)]

Mary-Obj John-Subj loves: λ V[V[(Mary′)(John′)]](love) $\Rightarrow$
love′(Mary′)(John′)

d. OVS:

Mary-Obj: λ **P**[λ V(Mary′)(**P**)]]

Mary-Obj love: λ **P**[λ V(Mary′)(**P**)]](love′) $\Rightarrow$
λ **P**[love′(Mary′)(**P**)]

Mary-Obj loves John-Subj: λ **P**[love′(Mary′(**P**)](John′) $\Rightarrow$
love′(Mary′)(John′)

e. VSO:

love: λ **P**[λ **Q**[love′(**Q**)(**P**)]]

John-Subj love: λ **P**[λ **Q**[love′(**Q**)(**P**)]](John′) $\Rightarrow$
λ **Q**[love′(**Q**)(John′)]

John-Subj love Mary-Obj: λ **Q**[love′(**Q**)(John′)](Mary′) $\Rightarrow$
love′(Mary′)(John′)

f. VOS:

love λ **Q**[λ **P**[love′(**Q**)(**P**)]]

Mary-Obj love: λ **Q**[λ **P**[love′(**Q**)(**P**)]](Mary′) $\Rightarrow$
λ **P**[love′(Mary′)](**P**)

Mary-Obj love John-Subj: λ **P**[love′(Mary′)(**P**)](John′) $\Rightarrow$
love′(Mary′)(John′)

We will assume that the sequence of semantic combination follows the exact syntactic order, but does not follow the order which the formal logicians have set. The order of quantifying-in process will vary from language to language. SOV language will quantify-in its subject first, then the object, and then the verb will be instantiated. OVS language will quantify-in its object first, then the subject and then the verb will be instantiated. However, the result of the semantic operation will be identical despite the different syntactic order and different sequence of semantic interpretation. TVP con-

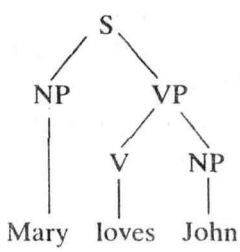
struction of Bach (1982), *take to task John,* will be assumed as the composition of (50b) instead of (50a).

(50) a. Embedding: Mary takes ___ to task + John
b. Concatenation: Mary takes John ___ + to task

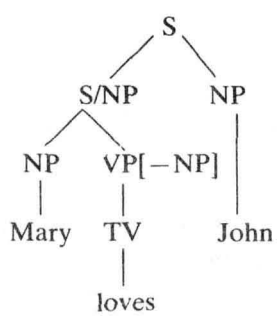
The sequence of semantic interpretation of GPSG is commonly considered bottom-up, from right to left, following the convention of the categorial grammar. However, we will not follow the sequence of composition of either GPSG or Pollard(1984). We will simply adopt a linear-ordered sequence of composition: in the case of English, the subject NP first and V next, etc. Our way of syntactic and semantic operation is done on the principle of **first-come-first-served.**

The current approach of linear-ordered operation faces the problem of categorization. A category can not combine with a noncategory, nor a noncateogry with a category. In the following tree diagram (51a), a syntactic node NP, according to the conventional operation, is not allowed to combine with V. Our linear-ordered operation obliges us to adopt the operation of (51b). Following this line of argument, a new kind of syntactic or a semantic category should be given to the result of the combination of NP with its immediately following node TV. And we assign to the result of NP-V combination a category S[−NP] as in (51b), since the node requires NP to become an S. This line of solution will not lose cross-categorical generalization nor cross-structural generalization with the help of two category features: *to be filled* and *filled.*

(51) a.



b.



This approach may explode the number of categories, but it does not cost much since the categorization is completely rule-governed when we consider a category is a set of features. *Mary loves* is a category S lacking an NP[ACC] or is an NP[NOM] combined with a VP lacking NP[ACC]. *Mary*

gave a book is also a category S lacking a PP[PFORM to]. Both syntactically and semantically we might as well follow the convention of Montague Grammar (Dowty *et al.* 1982:182), and recursively define the categories as:

(52) a. S is a category representing a sentence.
b. P is a category representing an NP term.
c. V is a category representing a V function.
d. If A and B are any categories, then A/B is a category.

Even though we follow this convention, its result is not fixed as in Montague grammar, but it is considered operational to adapt itself to the linearity in accordance with the syntactic order of a sentence. Despite such an operationality, the set of categories is completely predictable by a manageable number. The number of categories for lexical IV is only one with no additional categories. The subject NP is combined with lexcial IV lacking only a subject NP. By TV we will get only two categories; one itself, and another after it is combined with a subject NP. The latter requires combination with an object NP. And the number of the categories necessary for the linear translation is determinable.

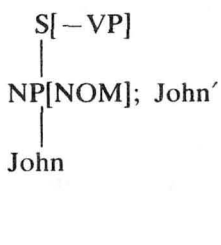
4. Applications on English Sentences

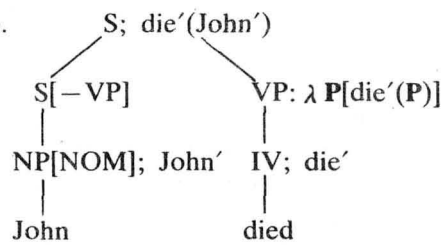
In this section we will apply our grammar on some basic sentences of English to show how it works, beginning with intransitive verbs, transitive verbs, ditransitive verbs, and then discontinuous constructions. In semantic representation, NP terms will be designated by the bold-faced capital letters **P, Q** and **R**; VP functors with V and W. Semantic translations of lexcial items will simply be represented with, on the right shoulder of a lexical item. NP terms will be considered as an argument; VP or IV will be considered as a function which takes an NP term to produce a sentential interpretation, following the convention of GKPS (1985; 188f).

4.1 Intransitive Verbs

Intransitive verbs are considered to have only one category, and no subcat number is assigned. VP generates IV by a unit production, in which there is only one daughter node, optional categories out of account. So VP is considered equivalent to IV.

(53) a. VP → IV
b. follow, rise, die, run. ... S[−NP[NOM]]; λ **P**[die′(**P**)]

(54) a.


b.


4.2 Transitive Verbs

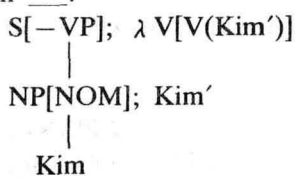
Phrasal category resulting from TV with NP is syntactically and semantically identical with an intransitive verb or a VP. Both can optionally combine with an adverb phrase. Both make a sentence or a sentential interpretation with a subject NP added. In other words, a transitive verb should be provided with NP[ACC] to become syntactically and semantically equivalent to IV or VP; a ditransitive verb needs NP[ACC] and NP[DAT] to become equivalent to IV; and so it goes on. The exposition of this line is still quite orthodoxical. But as we have already proposed, our operation should be a linear-ordered operation, and we need to make composition of the subject NP first with no regard to whatever category follows it. Combination of IV with NP[NOM] will simply result in a sentence. Traditionally a transitive verb is combined with an accusative NP to be a VP, which is in turn combined with a nominative NP to be a sentence. In LPSG the nominative NP is first combined with the transitive verb. The result of this linear ordered operation, NP[NOM] followed by a transitive verb, is assigned a syntactic category of S[−NP[ACC]] by default category assignment principle given in the previous section. S[−NP[ACC]] is then combined with NP[ACC] to be a sentence. Below we propose an ID rule and its corresponding syntactic category and semantic translation. Note that we sacrifice the sequence of lambda abstraction in the semantic expression. We propose that quantification begins with NP[NOM] followed by V and then by NP[ACC] to suit to the linear syntactic operation. This kind of schema has a by-product: we can avoid type-ambiguity which is caused by triple sisters (GKPS 1985: 194).

(55) a. VP → TV[1], NP[ACC]
b. kiss, love, close, ... TV, λ **P**[λ**Q**[kiss′(**Q**)(**P**)]]

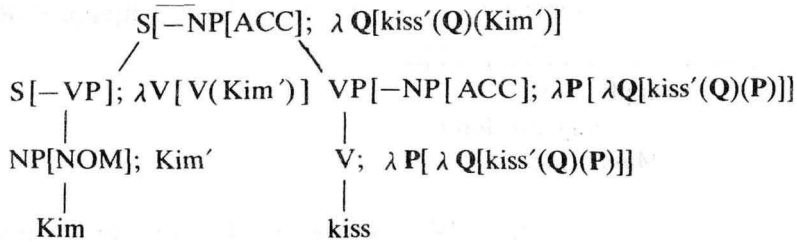
c. ___ kissed Lee S[−NP[NOM]], λ **P**[kiss″(Lee′)(**P**)]
d. Kim kissed ___ S[−NP[ACC]], λ **Q**[kiss″(**Q**)(Kim′)]

With NP[ACC] already provided, the result of translation in (55c) is semantically identical to that of IV which has only NP[NOM] abstracted. Translation of a transitive lexical item is double-abstracted. The process of combination for a simple example is given below:

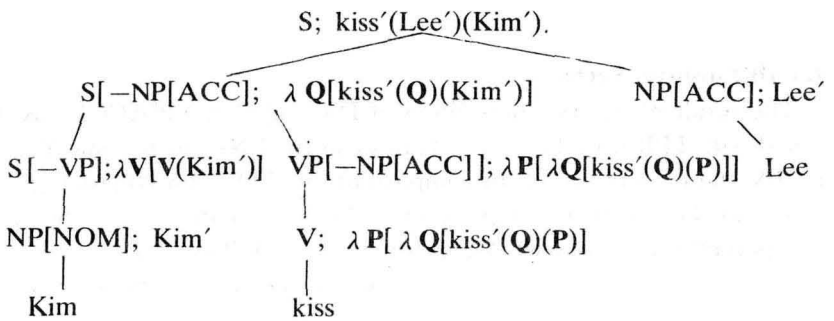
(56) a. Kim ___.



b. Kim kissed ___.



c. Kim kiss Lee.



4.3 PPV: Verb Requiring PP

There is a category of lexical PPV which requires PP to become a VP. For example, *apply* in (57) does not become an intransitive verb by itself without PP[to] following. In line with this, we construct ID rules by separating intransitive verbs which require PPs from normal ones. It seems to be a

misnomer to call the former either an intransitive or a transitive verb, since it differs from both in its way of composition: it needs PP to become IV and then NP[NOM] to become a complete S, which normal intransitive verbs need only NP, and TV needs two NPs to become a complete S. We will give a subcat number to distinguish what kind of PP they need to be followed by. Below we will give semantic translations only on the first rule. The other rules will be given identical translations, because the current translations only show functional relations between different constituents. We use NP terms of **P** and **R**, reserving **Q** for the type corresponding to NP[ACC]. The process of a semantic translations will be analogous to the translation of DTV[1].

(57) a. VP → PPV[1], PP[to] λ **P**[f(**R**)(**P**)],
b. appeal, apply, contribute .. PPV[1] λ**P**[λ **R**[appeal′(**R**)(**P**)]]
c. ___ appealed to John S[−NP[NOM]]; λ **P**[appeal′(John′)(**P**)]]
d. Mary appealed ___ S[−NP[ACC]]; [λ**R**[appeal′(**R**)(Mary′)]

(58) a. VP → PPV[2]; PP[for]
b. apply, sing, ...
c. ___ sang for John
d. Mary sang ___

There are five or more PPV subcategories. But we are not going to deal with them here.

4.4 Ditransitive Verbs

The result of binary composition of DTV[1] with NP[ACC] must be combined with PP[to] or PP[for] instead of normal NPs to become IV. We give a PPV status to the result of composition of DTV with NP[ACC], and a TV status to the result of composition of DTV with PP. The comment line in the parenthesis shows what constituent is to follow.
(;An opening parenthesis followed by ;designates a comment line)

(59) a. PPV[1] → DTV[1], NP[ACC] (; to be combined with PP[to])
λ **P**[λ **Q**[V(**R**)(**Q**)(**P**)]] (; PPV translation)
b. give, introduce,... DTV[1]
λ **P**[λ **Q**[λ **R**[give′(**R**)(**Q**)(**P**)]]]
c. ___ gave Fido ___ S[−NP][NOM], −PP[to]]
λ **P**[λ **R**[give′(**R**)(Fido′)(**P**)]]

d. ___ gave Fido to Sandy	S[−NP[NOM]] λ **P**[give′(Sandy′)(Fido′))**P**)]
e. John gave ___ ___	S[−NP[ACC], −PP[to]] λ **Q**[λ **R**[give′(**R**)(**Q**)(John′)]]
f. John gave Fido ___	S[−PP[to]] λ **R**[give′(**R**)(Fido′)(John′)]

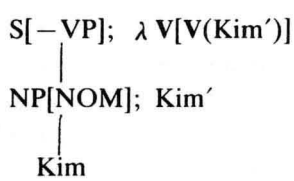
(60) a. TV →DTV[2], NP[DAT]

λ **P**[λ **R**[f(**R**)(**Q**)(**P**)]](; TV translation)

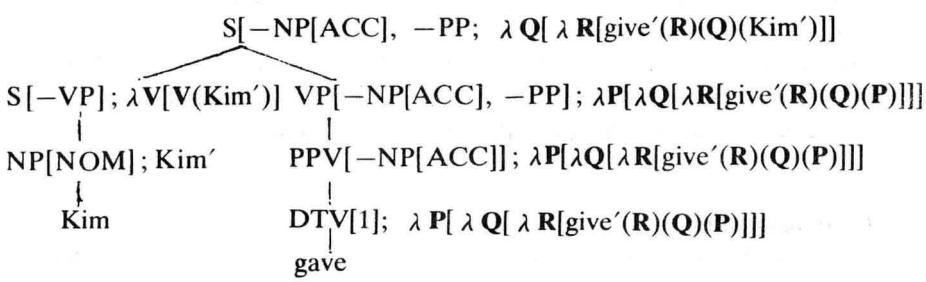
b. give, send, ...	DTV[2], λ **P**[λ **R**[λ **Q**[give′(**R**)(**Q**)(**P**)]]]
c. ___ gave Sandy ___	S[−NP[NOM], −NP[ACC], λ **P**[λ **R**[give′(Sandy′)(**R**)(**P**)]]]
d. ___ gave Sandy Fido	S[−NP[NOM]], λ **P**[give′(Sandy′)(Fido′)(**P**)]]]

The order of the two objects may be explained by LP statements of either NP < PP (for DTV[1]) or NP[DAT] < NP[ACC] (for the case of NP[ACC] before PP, while DTV[2] quantifies-in NP[DAT], which is semantically equivalent to PP, before NP[ACC]. The different semantic calculation is made possible by reversing the order quantifying-in operation in the case of DTV[2]: λ R first then λ Q. This is a radical departure from the semantics of GKPS (1985, 197), which advocates bottom-up and right to left semantic composition. The following is a process of homomorphic operation of example (59).

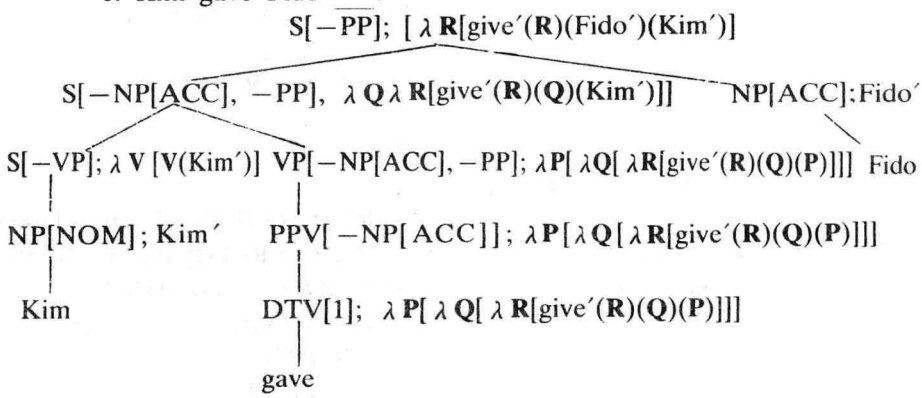
(61) a. Kim ___.



b. Kim gave ___.



c. Kim gave Fido ___.



d. Kim gave Fido to Sandy. (Sandy is simply quantified-in, giving us syntactic/ semantic representation of S, give′(Sandy′)(Fido′)(Kim′)])

What we have been trying to do is to establish a linear-ordered binary operation which is homomorphic between syntax and semantics. Our job, however, seems still far from being complete and convincing. But because of the limit of space we are going to skip other details and go over to unbounded depndency phenomenon.

5. Unbounded Dependencies

If an English grammar is to attain any linguistic significance, it should give generalizations on such constructions as active-passive relations, inversion and coordination unbounded dependencies. In this chapter, however, we will confine our concern to the unbounded dependencies. Unbounded dependencies had commonly been considered to be out of reach of any phrase structure grammar until Gazdar (1982a) expanded its power. He developed a way to explain it with derived categories using the concept of slash. We will accept the treatment by GKPS (1985) as it is, with FFP, HFC and CAP operating, and we will consider our mission completed by revising it to be suited to the linear-ordered binary grammar.

5.1 Topicalization

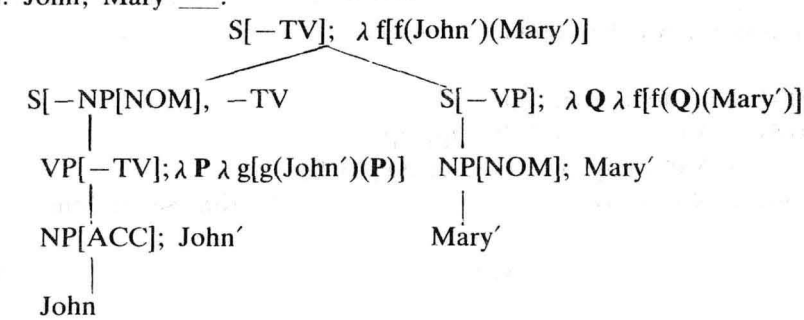
The S node lacks nothing until it is given an initial node. Provided with

an initial node of any kind, ID rule default is activated to introduce lacked categories, which should be satisfied somewhere in the tree, keeping well--formed conditions such as LP statement. Our first example is a simple sentence beginning with topicalized NP[ACC] as *John Mary loves.* Its syntactic operation was already shown in section three, and we will be concerned mainly on how the semantics operate.

(62) a. John ___

S[−NP[NOM], −TV]
|
VP[−TV]; λ **P** λ g[g(John′)(**P**)]
|
NP[ACC]; John′
|
John

b. John, Mary ___.

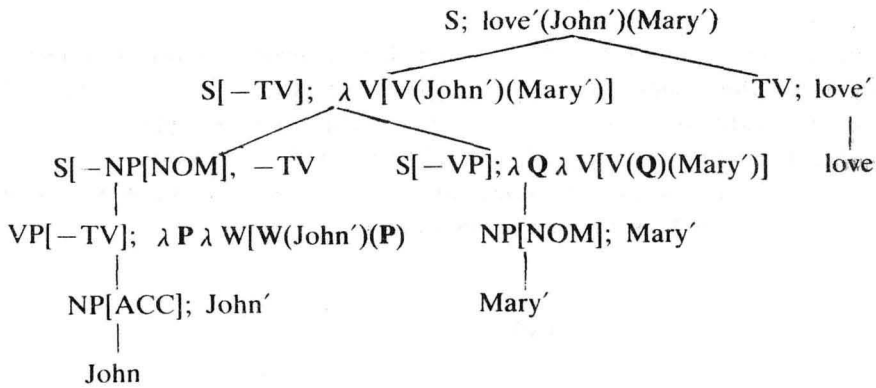


The lambda calculation of S[−TV], i.e. on the composition of *John* and *Mary* is as follows:

(63) λ **P**[λ W[W(John′)](**P**)](λ **Q** λ V[V(**Q**)(Mary′)])
=> λ W[W(John′)](λ **Q** λ V[V(**Q**)(Mary′)])
=> λ V[V(John′)(Mary′)]

And the result of the operation is as simple as:

(62) c. John Mary loves.



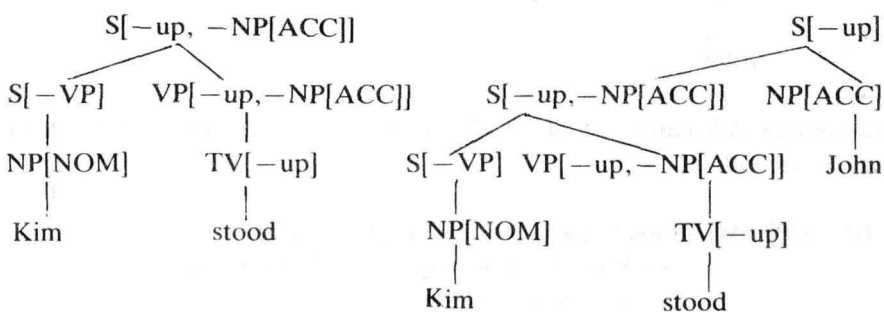
5.2 Crossing

Embedded construction causes no problem with our binary grammar as we saw in section one, for it is easily manipulated by a CF grammar. Crossing construction, however, seems to deserve our further concern. Here our concern will be confined in syntax.

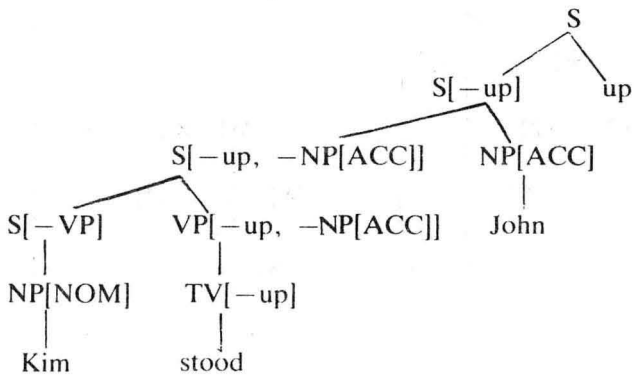
(64) Kim stood John up.

(65) a. VP → VP[−up] up
b. VP[−up] → TV[−up] NP[ACC]

(66) a. Kim stood ___. b. Kim stood John ___.

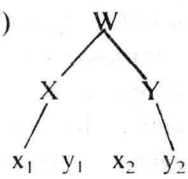


c. Kim stood John up.



We will now see how our binary grammar deals with the double crossing constructions as given below. Based on the tree (67) we can construct ID rules and LP statements as in (69) and (70).

(67)

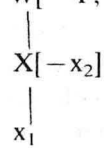


(68) a. $W \rightarrow X, Y$
b. $X \rightarrow x_1, x_2$
c. $Y \rightarrow y_1, y_2$

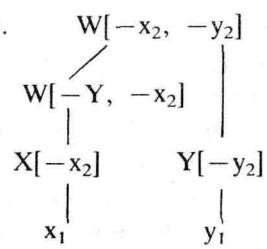
(69) LP statements:
a. $X < Y$
b. $x_1 < y_1 < x_2 < y_2$

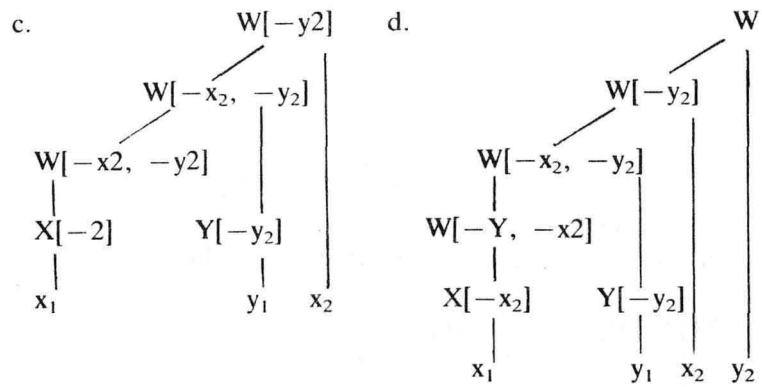
ID rules in (68) and LP statements in (69) will give us the following tree with the help of ID rule default:

(70) a.



b.





6. Conclusion

Our current work began with the belief that the latest version of GPSG is defective in that it does not provide us with homomorphic operation between syntax and semantics, and in that its semantic operation is not context-free and linear-ordered (GKPS 1985: 197f). Our assumption in this paper is that linearity is crucial for a semantic theory as well as for a syntactic theory if it is to be psychologically realistic. We have been trying to revise the GPSG into a binary grammar so that both syntactic and semantic operations are homomorphic and linear-ordered from left to right.

At the bottom of this whole revision lies the expansion of the expressiveness of the category by default value assignment based on the ID rule defualts. The formalization is complicated, but it is still a context free grammar. And furthermore it does not imply any complication of the process of both generation and recognition, for it is possible to devise a CF grammar which is nearly as simple as a regular grammar and which is operable in left-to-right linear operation as well as homomorphic between syntax and semantics.

The current binary linear-ordered grammar is far from being complete as is presented in this paper. It does not say much about some typical issues such as the complex NP constraint, the left branching constraint, parasitic gaps, and the double hole constraint. We merely dealt with a few examples to show its expressive power. The semantic type assignment on the syntactic categories, as we see in the note on chapter three, is nothing but an extemporaneous attempt. The most seriously lacking is constraining devices on the operation of the ID rule default. The above theorization is far from being

complete and convincing. And our belief is that the most difficult task in constructing the linear-ordered grammar seems to be establishing a convincing system of categorization.

REFERENNCES

Aho, A. V. and J. D. Ullman (1972) *The Theory of Parsing, Translation and Compiling,* Prentice-Hall, Englewood Cliffs, NJ.

Aravind, K. J. (1985) 'Tree Adjoining Grammars,' in D. R. Dowty et al., eds., *Natural Language Parsing,* Cambridge University Press, London.

Bach, Emond (1970) 'Pronominalization,' *Linguistic Inquiry* 1, 121-122.

_____ (1973) *Syntactic Theory,* Holt, Rinehart and Winston, NY.

_____ (1982) 'Purpose Clause and Control,' in P. Jacobson and G. K. Pullum, eds., *The Nature of Syntactic Representation,* D. Reidel, Dordrecht, Holland.

Barwise, J. and S. Peters (1987) *Lectures on Situation Theory and Situation Semantics,* Mimeograph.

Bear, J. (1982) *Gaps as Syntactic Features,* IULC.

Brame, M. (1979) *Essays toward Realistic Syntax,* Noit Amrofer, Seattle, WA.

Charniak, E., and D. McDermott (1984) *Introduction to Artificial Intelligence,* Addison-Wesley, Reading, MA.

Chomsky, N. (1957) *Syntactic Structures,* Mouton, The Hague.

_____ (1965) *Aspects of the Theory of Syntax,* The MIT Press, Cambridge, MA.

_____ (1970) 'Remarks on Nominalization,' in R. A. Jacobs and P. S. Rosenbaum, eds., *Readings in English Transformational Grammar,* Ginn and Company, Waltham, MA.

_____ (1975) *Logical Structure of Linguistic Theory,* Plenum, NY.

Church, K. W. (1979) *Phrase Structure Parsing,* IULC.

Dowty, D. R., L. Karttunen, and S. Peters (1982) *Introduction to Montague Semantics,* Reidel, Dordrecht, Holland.

Fordor, Janet D. (1983) 'Phrase Structure Parsing and the Island Constraints,' *Linguistics and Philosophy* 6, 163-224.

Gazdar, G. (1982a) 'Coordinate Structure and Unbounded Dependenices,' in Barlow M., D. P. Flickinger and I. A. Sag, eds., *Developments in Generalized Phrase Structure Grammar,* IULC.

_____ (1982b) 'Phrase Structure Grammar,' in P. Jacobson and G. K.

Pullum, eds., *The Nature of Syntactic Representation,* Reidel, Dordrecht, 131-186.

_____, E. Klein, G. Pullum and I. Sag (1985) *Generalized Phrase Structure Grammar,* Harvard Univ. Press, Cambridge, MA.

Groenendijk, J. A. G., T. M. V. Janssen and M. B. J. Stokhof (1981) *Formal Methods in the Study of Language,* Holland.

Gross, M. (1972) *Mathematical Models in Linguistics,* Prentice-Hall, Englewood Cliffs, New Jersey.

Harman, G. (1963) 'Generative Grammar without Transformation Rules,' *Language* 39, 597-616.

Harris, M. D. (1985) *Introduction to Natural Language Processing,* Prentice-Hall, Englewood Cliffs, New Jersey.

Hopcroft, J. E., and J. D. Ullman (1969) *Formal Languages and Their Relation to Automata,* Addison-Wesley, Reading, MA.

_____ (1979) *Introduction to Automata Theory, Languages, and Computation,* Addison-Wesley, Reading, MA.

Huck, G. J. (1985) *Exclusivity and Discontinuity in Phrase Structure Grammar,* University of Chicago.

_____ and A. E. Ojeda (1987) *Syntax and Semantics: Discontinuous Constituency,* Academic Press, Orlando, Florida.

Jackendoff, R. (1977) 'Constraints on Phrase Structure Rules,' in P. W. Culicover, T. Wasow and A. Akmajian, eds., *Formal Syntax,* Academic Press, NY. 71-132.

Langendoen, D. T. (1975) 'Finite State Parsing of Phrase Structure Languages and the Status of Readjustment Rules in the Grammar,' *Linguistic Inquiry* 6, 533-554.

_____ (1977) 'On the Inadequacy of Type-3 and Type-2 Grammars for Human Languages,' in P. J. Hopper, ed., *Studies in Descriptive and Historical Linguistics: Festschrift for Winfred P. Lehmann,* Amsterdam, Holland, 159-171.

Marcus, M. P. (1981) *A Theory of Syntactic Recognition for Natural Language,* MIT Press, Cambridge, MA.

Peters, S. and R. Ritchie (1973) 'On the Generative Power of Transformational Grammar, *Information Sciences* 6, 49-84.

Pollard, C. J. (1984) *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language,* Ph. D. dissertation, Stanford Univ.

_____ (1985a) *Lectures on HPSG,* Mimeograph.

_____ (1985b) *Phrase Structure Grammar without Metarules,* Mimeograph.

Pollard, C. J. and I. Sag. (1987) *Information-based Syntax and Semantics,*

Vol. 1, CSLI, Stanford University.

Pullum, G. K., and G. Gazdar (1982) 'Natural Languages and Context-free Languages,' *Linguistics and Philosophy* 4, 471-504.

Reich, P. A. (1969) 'The Finiteness of Natural Language,' *Language* 45, 831-843.

Ritchie, R. W. (1973) 'Context-sensitive Immediate Constituent Analysis: Context-free Language Revisited,' *Mathematical Systems Theory* 6, 324-333.

Schank, R. C. and K. M. Colby (1973) *Computer Models of Thought and Language*, Freeman, San Francisco.

Sells, P. (1985) *Lectures on Contemporary Syntactic Theories*, CSLI, Stanford, CA.

Sheil, B. A. (1976) 'Observations on Context Free Parsing,' *Statistical Methods in Linguistics* 7, 71-109.

Shin, Gyonggu (1987) *A Phrase Structure Approach to English Syntax*, Ph. D. dissertation, Chonbuk National University, Korea.

Valiant, L. G. (1975) 'General Context-free Recognition in Less than Cubic Time,' *Journal of Computer and System Sciences* 10, 308-315.

Wall, R. (1972) *Introduction to Mathematical Linguistics*, Prentice-Hall, Englewood Cliffs, NJ.

Winograd, T. (1972) *Understanding Natural Language*, Academic Press, NY.

_____ (1984) *Languages as a Cognitive Process*, Addison-Wesley, MA.

Woods, W. (1970) 'Transition Network Grammars for Natural Language Analysis,' *Comm. ACM.* 13, 591-606.

Department of English
Chonnam Nat'l University
Yongbong-dong, Buk-ku,
Kwang-Ju, 500-757
Korea