JPSG—A Phrase Structure Grammar for Japanese*

HARADA, Yasunari
Faculty of Law
Waseda University
1-6-1 Nishi-Waseda, Shinjuku-ku
Tokyo 160, Japan

GUNJI, Takao
Faculty of Language and Culture
Osaka University
1-1 Machikaneyama-cho, Toyonaka
Osaka 560, Japan

MIYOSHI, Hideo
Information System Laboratory
Sharp Corporation
Minosho-cho, Yamato-Koriyama-shi
Nara 639-11, Japan

SIRAI, Hidetosi
Department of Information and Communication Engineering
Tamagawa University
6-1-1 Tamagawa-Gakuen, Machida-shi
Tokyo 194, Japan

HASIDA, Kôiti
Second Laboratory
Institute for New Generation Computer Technology
1-4-28 Mita, Minato-ku
Tokyo 108, Japan

October 1, 1988

Abstract

This article describes a phrase structure grammar for Japanese called JPSG (Japanese Phrase Structure Grammar), and a parser based on this grammar. JPSG is a grammar formalism for the Japanese language based on the recent development of phrase structure grammar theories embodied in such frameworks as GPSG (Generalized Phrase Structure Grammar) and HPSG (Head-driven Phrase Structure Grammar), with certain extensions relevant for the description of Japanese. Especially, the extension of the **subcat** feature facilitates the description of word order variation in Japanese. This extension also allows to capture wider range of generalizations over phrase structure rules in Japanese than traditional phrase structure approaches. Since JPSG belongs to the class of the so-called unification-based grammars, it could also serve as a convenient grammatical system from the point of view of computational linguistics. A parser that implements its characteristics in a natural way is currently being developed in a variant of logic programming language.

1 Introduction

In a natural language understanding system, grammar is as important a component as knowledge bases (such as the world knowledge or inference rules based on 'common sense'), and how it is organized may affect the overall performance of the system. A computational grammar formalism based on a sound linguistic theory serves as a useful base for many application systems of natural language processing.

We are in the process of developing a phrase structure grammar for Japanese called JPSG (Japanese Phrase Structure Grammar) [5], incorporating some of the basic concepts embodied in GPSG (Generalized Phrase Structure Grammar) [2], and its descendant HPSG (Head-driven Phrase Structure Grammar) [10]. A parser is also being implemented based on the grammatical foundations of JPSG.

GPSG is a grammatical theory that started as an extension of CFG (context free grammar). As the theory is developed, many of the syntactic regularities come to be expressed as grammatical principles, not as phrase structure rules. This reduction of the role of specific rules in the grammar has further been pursued in HPSG, resulting in more abstract forms of phrase structure rules. The interpretation of these abstract phrase structure rules largely depends on the particular lexical items appearing in the phrase structure tree and the interaction of them with various general principles concerning certain specific features within categories.

JPSG, while retaining most of the universal principles that are utilized in these frameworks, is further extended so that it could cope with word order variation in Japanese. At the same time, since JPSG is a unification-based grammatical formalism as GPSG and HPSG, it is best suited for logic programming language implementation. In this paper, we will describe the basic ideas of JPSG as well as operating principles of its parser implementation. In the next section, major characteristics of the theory of phrase structure grammar will be briefly reviewed. Following the sketch of the JPSG theory in Section 3, its implementation will be discussed in Section 4.

2 Phrase Structure Grammars

The theory of phrase structure grammar is a relatively recent development (in fact, reconstruction) in generative grammar. As the first formalized system of the modernized phrase structure grammar, GPSG has attempted to describe every linguistic phenomena, most of which have traditionally been described through postulation of transformational rules, within the framework of context free grammar. Since transformations have been known to be the source of both theoretical and computational difficulties, a theory based on such a constrained framework has since been gaining attention of both linguistic and computational circles. Although the formulations of GPSG has undergone some minor changes in the past several years, its fundamental principles remain the same. We list some of the basic characteristics of GPSG below:

(2.1) a. *Feature-Based*

Grammatical categories are construed not as monadic symbols but as complex symbols with internal structures. In [2], they are defined as sets of *features* (pairs of feature name and feature values).

b. *Monostratal*

Only one kind of grammatical structure is used in the grammatical description.

The single structure represents every grammatical information. As the consequence of this, transformational rules are never used.

c. *Metagrammatical*
Instead of transformational rules, the relationships between phrase structures are captured at the metagrammatical level in terms of *metarules.*

d. *Principle-Based*
Generalizations about phrase structure rules are also captured by general principles on possible phrase structures. These principles express the constraints on proper distribution of features in a local branching of phrase structure trees.

e. *Semantics-Oriented*
Syntax and semantics are closely linked by way of Montague semantics [8].

As an extension of GPSG, HPSG shares many of the basic ideas, while introducing a different set of features and associated principles. One of the most remarkable difference is in the definition of the feature on subcategorization. While in GPSG the subcategorization feature refers to the corresponding bar-levels of a category in the projection hierarchy, HPSG assigns as the value of the subcategorization feature a list of syntactic categories of the complements which the head of the phrase (e.g. the verb in a verb phrase) requires. This enables one to express a wider range of phrase structures in terms of a single rule, and a variety of linguistic phenomena are explained through a few phrase structure rules.

HPSG has the following additional or revised properties:

(2.2) a. *Lexical*
The relationships between lexical items are captured at the level of lexicon in terms of *lexical rules.* (cf. (2.1c) above)

b. *Semantics-Oriented*
Syntax and semantics are closely linked by way of Situation Semantics [1]. (cf. (2.1e) above)

The use of lexical rules in HPSG, instead of metarules in GPSG, sets the mathematical property of the grammar more restricted, since the lexical rules are operations on a finite domain, while metarules, on phrase structure rules (ID rules), may not.[1]

As for the use of Situation Semantics, rather than Montagovian model-theoretical semantics, the net effect is not yet clear. However, a semantic theory whose basic operation is functional application (namely, Montague semantics), seems to be far more restricted than a so-called 'relational' semantic theory in that partiality of information is not assumed and hence the relationships between pieces of partial information are not adequately handled. Moreover, since the basic grammatical operation in recent trends in phrase structure grammar is *unification*, a relational theory could be more easily incorporated in the grammatical system.

These theories of phrase structure grammar provide us with a convenient framework to describe various linguistic phenomena that are found in natural language. However, the fact that these grammatical theories have been developed with the description of English

[1]In fact, the metarules in GPSG are severely restricted by the so-called Lexical-Head Constraint, which restricts the domain of application to rules introducing lexical items, thus effectively reducing the domain of application to a finite set.

in mind has led us to the need for further extending the overall framework. Even though these theories of phrase structure grammar are all intended to be a basis for a 'universal grammar', we are not content with the current versions.

One of the motivation for the need of an extension is the fact that the Japanese language shows a marked uniqueness in its relative freedom of word order, while word order is relatively fixed in English. This fact has led us to rethink the nature of complements of a head in JPSG. We believe that, at least for the description of Japanese, the information on subcategorization and that on word order are independent of each other. Unlike HPSG, we assume that the complements as the value of the subcategorization feature have no inherent ordering. This will enable us the proper handling of word order variation (see the discussion of **subcat** in the next section).

Thus, the characteristics of JPSG can be summarized in the following way.

(2.3) a. *Feature-Based*
As with GPSG and HPSG, grammatical categories are construed as complex symbols with internal structures.

b. *Monostratal*
As with GPSG and HPSG, JPSG is a monostratal theory of grammar, with no transformation.

c. *Rule-Independent (Lexical and Principle-Based)*
The canonical syntactic structure in Japanese is stated by a single phrase structure rule. As the consequence, the phrase structure rule itself hardly expresses specific grammatical information. Instead, most of the grammatical information is expressed either in the lexicon or by general principles on the distribution of features in the phrase structure. Most of the principles that are employed in GPSG and/or HPSG are retained in our system.

d. *Integrated*
Morphological, syntactic, semantic, and possibly pragmatic, pieces of information are expressed in an integrated representational system, thus achieving both modularity and interconnectedness of these components. As with HPSG, we have been examining a modified version of Situation Semantics for the semantic representation.

The next section will elaborate on each of these points. A brief survey and comparison among transformational grammar, GPSG, HPSG, and JPSG can be found in [3].

3 Grammar

To get the general idea of how phrase structure grammar in general, and JPSG in particular, is organized, we start with the discussion of the following concepts: *feature, phrase structure rule*, and *grammatical principle*. We will elaborate on each point with some examples.

3.1 Features

Grammatical categories are represented as sets of features. Features can be grouped into the following three according to the form of the values they take.

(3.1) *Binary Feature*: A feature that takes either + or − as its value is called a *binary feature.*

(3.2) An Example of a Binary Feature

pas (passivizable) designates whether or not a verb can be passivized.

This feature is used to block multiple passivization. If the **pas** value of a verb phrase is + due to an application of passivization, it cannot be further passivized.

(3.3) *Multi-Valued Feature*: A feature that takes one element of a predetermined set as its value is called a *multi-valued feature.*

(3.4) Examples of Multi-Valued Features

pos (part of speech) takes the value from $\{v,\ n,\ p,\ adv,\ adn,\ \ldots\}$ and designates the part of speech of the grammatical category involved.

pform (postposition form) takes one of $\{ga,\ wo,\ ni,\ no,\ de,\ \ldots\}$ as the value and designates the kind of postposition involved.

gr (grammatical relation) takes either *sbj* or *obj* as the value and differentiates a subject phrase from an object phrase.

sem (semantics) designates the semantic representation of the grammatical category involved. The exact form of the value depends on the semantic theory adopted.

(3.5) *Category-Valued Feature*: A feature that takes as its value a set of categories is called a *category-valued feature.*

There are three category-valued features in our current system.[2]

(3.6) Examples of Category-Valued Features

subcat (subcategorization) designates the set of categories (complements) that a particular category (head) requires. For example, the value of **subcat** for the verb *aisitei* 'love' is specified as {p[*sbj*], p[*obj*]}, which means that this verb takes a subject postposition phrase (p[*sbj*]) and an object postposition phrase (p[*obj*]).

slash ('/') designates a syntactic gap within the grammatical category involved. This is indispensable in handling unbounded dependency constructions such as topicalized sentences and relative clauses. See [5, Chap. 5] for the analysis of unbounded dependency constructions in Japanese.

refl (reflexive) designates whether or not the category dominates an occurrence of *zibun* 'self'. This takes as its value either ϕ (empty set) or the singleton set {p[*sbj*]}. In the latter case, the phrase is marked as dominating *zibun* and the p[*sbj*] in the value of **refl** is utilized in the binding of the reflexive pronoun. We will see an example in Subsection 3.5

[2]In this paper, categories are designated by a left square bracket ("[")followed by an indefinite number of feature specifications (a feature name followed by its value) separated by commas(",") followed by a right square bracket ("]"). When the value uniquely determines the name, the name can be omitted. Also, when the value is ϕ or not relevant, the entire feature specification can be omitted. Finally, a category of the form [**pos** *c*, ..., **sem** *s*] is often abbreviated as *c*[...]:*s*.

The **subcat** feature in JPSG is an extension of the SUBCAT feature in HPSG, which takes an ordered list of categories. The **subcat** feature in JPSG, on the other hand, takes as its value an unordered set rather than an ordered list of categories: thus there are no inherent ordering among the elements of the value of **subcat**. The subcategorized-for categories in HPSG, on the other hand, are ordered according to the grammatical obliqueness; the subject (least oblique) comes last, then the direct object, and so on. This ordering is coupled with one of the linear precedence constraints for English to arrange the surface order of the complements in English. For example, if there are more than one complements, the least oblique complement comes at the leftmost position. Thus, you have *give Ken a book*, but not **give a book Ken* (*Ken*, being the direct object, is less oblique than the indirect object *a book*) [10, Chap. 7]. As mentioned above, our system expresses the grammatical relation directly with a separate feature **gr**. Thus, the ordering in the **subcat** value has no significance. It is also the fact of Japanese that grammatical relations plays no role in ordering complements, since, as seen in the next subsection, we assume only a binary-branching phrase structure rule, with at most one complement at the same time for a head. Moreover, if a verb subcategorizes for several complements at distinct hierarchical levels (for example, a subject and an object), these complements can appear in any order.

(3.7) a.	Naomi-ga	Ken-ni	hon-wo	ageta.
	NOM	DAT	book-ACC	gave
b.	Naomi-ga	hon-wo	Ken-ni	ageta.
	NOM	book-ACC	DAT	gave
c.	Ken-ni	Naomi-ga	hon-wo	ageta.
	DAT	NOM	book-ACC	gave
d.	Hon-wo	Naomi-ga	Ken-ni	ageta.
	book-ACC	NOM	DAT	gave
e.	Ken-ni	hon-wo	Naomi-ga	ageta
	DAT	book-ACC	NOM	gave
f.	Hon-wo	Ken-ni	Naomi-ga	ageta
	book-ACC	DAT	NOM	gave

'Naomi gave Ken a book.'

In the first sentence above, the verb *ageta* 'gave' takes the accusative object *hon-wo* 'a book' as the first complement, then the dative *Ken-ni*, and finally the subject *Naomi-ga*. On the other hand, the same verb in the second sentence takes the dative object *Ken-ni* as the first complement, and then the accusative and the subject. The rest of the sentences show other possibilities. These facts could not possibly be handled adequately if we assumed a fixed order among the complements in the value of the **subcat** feature. See [4] and [5, Chap. 6] for more discussion on the interaction between subcategorization and word order variation.

Using these features, traditional category symbols such as S (sentence), VP (verb phrase), TVP (transitive verb phrase), and PP (postposition phrase) are expressed as a set of features in the following way:

(3.8) a. S: [**pos** *v*, **subcat** ϕ]

b. VP: [**pos** *v*, **subcat** {p[*sbj*]}]

c. TVP: [**pos** *v*, **subcat** {p[*sbj*], p[*obj*]}]

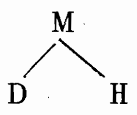


Figure 1: Fundamental Phrase Structure in Japanese

d. PP: [**pos** *p*, **subcat** ϕ]

Likewise, entries in the lexicon have exactly the same kind of feature structure:[3]

(3.9) a. *Ken*: [**pos** *n*, **subcat** ϕ, **sem** *ken*]

b. *aruk*: [**pos** *v*, **subcat** {p[*sbj*]:X}, **sem** *walk*(X)]

c. *aisitei*: [**pos** *v*, **subcat** {p[*sbj*]:X, p[*obj*]:Y}, **sem** *love*(X,Y)]

3.2 Phrase Structure Rule

In JPSG, only one phrase structure rule is postulated, namely the following:

(3.10) *Phrase Structure Rule*
$M \rightarrow D\ H$

where M stands for an arbitrary mother category, D a daughter category, and H a head category.

The phrase structure rule (3.10) expresses that the fundamental syntactic structure of Japanese is represented by the binary branching tree as shown in Fig. 1. That is, a number of phrase structure rules, such as S $\rightarrow$ PP VP, VP $\rightarrow$ PP V, etc., which have often been assumed in natural language processing systems for Japanese, are reduced to a single rule with the head category at the right position. This reduction of phrase structure rules has been made possible by treating grammatical categories as sets of features and stating syntactic generalities in the form of grammatical principles.

We are assuming three basic constructions for Japanese: *complementation*, *adjunction*, and *coordination*:

(3.11) a. *Complementation*
$M \rightarrow C\ H$

b. *Adjunction*
$M \rightarrow A\ H$

c. *Coordination*
$M \rightarrow H\ H$

[3]The semantic representation below, in the form of predicate logic, is only for the expository purpose; actual representations will become much more complicated using the primitives based on Situation Semantics [1].

where C stands for a complement and A an adjunct.

We will concentrate on the formulation of complementation in this paper, relegating the formulation of adjunction and coordination, which are under way, possibly to a future report in a separate occasion.

In the following subsections, we will discuss three major grammatical principles, showing how the reduction of phrase structure rules in the form of (3.11) is made possible.

3.3 Head Feature Principle

It is often the case that most of the grammatical properties of a phrase are merely the reflection of the head of the phrase. For example, a verb phrase has the verbal properties because the head of the phrase is a verb. Thus, the values of many features are shared by the head-category and the mother category in a local tree. In other words, the values *unify* with each other. This general principle has been formulated as the HFC (Head Feature Convention) in GPSG and as the HFP (Head Feature Principle) in HPSG and JPSG. The general idea behind these formulations is to state certain constraints regarding the *head features.* The following gives the general idea of this principle:

(3.12) *Head Feature Principle*
The value of a head feature at the mother category (in a given local tree) unifies with the value of the head feature at the head category.

Among the features in JPSG mentioned so far, features other than **sem**, **subcat**, **slash**, **refl** are head features. That is, the head features include **pas**, **pos**, **pform**, and **gr**. This principle guarantees that the values of these head features at the M node and those at the H node in Fig. 1 unify with each other. Fig. 2 shows how this principle works in a simple sentence. In example (a), since the VP is the head category of the sentential phrase, the value of the **pos** feature at the VP node and the value of **pos** at the S node unify with each other and has the value *v*. Example (b) shows the internal structure of the subject phrase. In this case, in addition to the value of **pos**, that of **gr** are shared between the mother and the head.

3.4 Subcat Feature principle

The Subcat Feature Principle, or the Subcategorization Principle as is called in HPSG, states the relationship that holds between the **subcat** value of the mother and that of the head in a given local tree. As described in Subsection 3.1, **subcat** in JPSG takes as its value a set of categories that would count as complements of the head. Thus, our formulation mentions no order-specific status of an element of the **subcat** value. It is stated as follows:

(3.13) *Subcat Feature Principle*
In the phrase structure rule in (3.10), the value of **subcat** at the M category unifies with that obtained by subtracting the D category from the value of **subcat** at the H category.

A simple example of this is also seen in Fig. 2 (a). Since the VP *aruk* 'walk' is an intransitive verb, its **subcat** value is {p[*sbj*]}, a singleton set consisting of a subject postposition phrase. This unifies with the p[*sbj*] that precedes *aruk*, namely the postposition

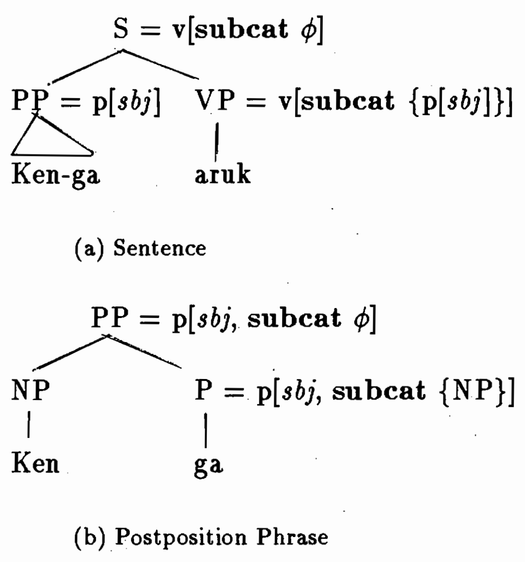

S = v[**subcat** ϕ]
PP = p[*sbj*]
VP = v[**subcat** {p[*sbj*]}]
Ken-ga
aruk
(a) Sentence
PP = p[*sbj*, **subcat** ϕ]
NP
P = p[*sbj*, **subcat** {NP}]
Ken
ga
(b) Postposition Phrase

Figure 2: Examples of the Head Feature Principle and the Subcat Feature Principle

phrase dominating *Ken-ga*. Thus, according to the Subcat Feature Principle, the **subcat** value of the mother is ϕ. A similar situation can also be seen in example (b).

By formulating the Subcat Feature Principle as above, we can handle the word order variation observed in Japanese only by postulating the phrase structure rule (3.10). Fig. 3 shows an example of such a case. The tree diagram in (a) shows the phrase structure tree for the 'unscrambled' canonical sentence *Ken-ga Naomi-wo aisiteiru* 'Ken loves Naomi', while the diagram in (b) represents the phrase structure tree for the 'scrambled' sentence *Naomi-wo Ken-ga aisiteiru*. In either case, the Subcat Feature Principle and the phrase structure rule in (3.10) suffice to sanction these phrase structures and all the V categories are assigned the same **sem** value, namely *love*(*ken*, *naomi*).

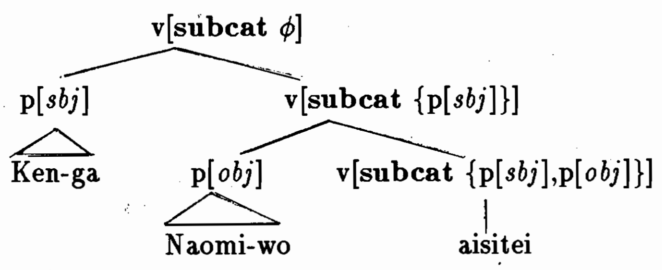
3.5 Binding Feature Principle

This principle states the distribution of binding features over a given parse tree. A *binding feature* is a feature whose value is determined with respect to a category possibly separated by a number of sentence boundaries. In other words, this feature often plays a crucial role in describing the so-called *unbounded dependency* phenomena such as topicalization and relativization. A familiar example of unbounded dependency in English is the relationship between a fronted *wh*-phrases at the sentential-initial position and the 'gap' corresponding to the *wh*-phrase:

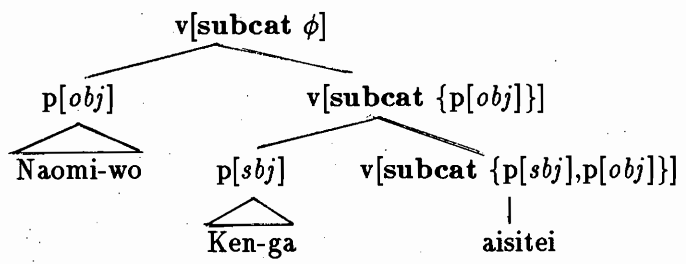
(3.14) a. Which students did the professor believe that Naomi said *e* are coming?

b. *Which students did the professor believe that Naomi said *e* is coming?

Note that the gap (designated by *e*) must bear the information on the number (plural in this case) corresponding to the *wh*-phrase *which students*, since the verb form must be plural (*are* and not **is*).



(a) Canonical Word Order



(b) 'Scrambled' Word Order

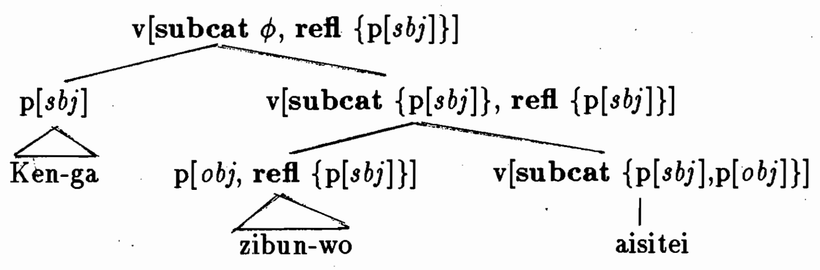
Figure 3: Analysis of 'Scrambling'

In JPSG, two *binding* features are postulated to bear such unbounded information, namely **slash** and **refl**. The former involves the description of topicalization and relativization just as in English, while the latter reflexivization. The Binding Feature Principle, or the Foot Feature Principle as is called in GPSG and the Binding Inheritance Principle as is called in HPSG, is formulated as follows:

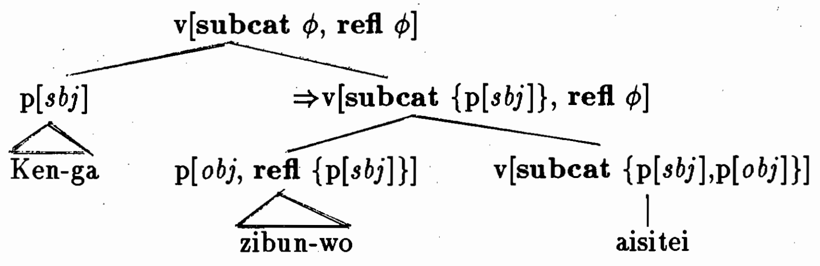
(3.15) *Binding Feature Principle*
In the phrase structure rule in (3.10) the value of a binding feature at the M category unifies with the union of its value at the D category and its value at the H category minus the category bound at this local branching.

An example where the Binding Feature Principle is applied to **refl** is shown in Fig. 4, where both (a) and (b) are phrase structure trees for the string *Ken-ga zibun-wo aisiteiru* 'Ken loves himself'. In our formulation of reflexivization, it is assumed that if there is an instance of a p[*sbj*] within the **subcat** value of a category that dominates an occurrence of *zibun*, this p[*sbj*] can bind the *zibun*, in the sense that this p[*sbj*] can unify with the p[*sbj*] in the value of **refl**. According to the Binding Feature Principle, when *zibun* is bound, the value of **refl** is not propagated upward beyond that category. See [5, Chap. 4] for further details of this analysis.

Fig. 4 (a) shows the case in which the **refl** value is propagated up to the topmost node, in which case *zibun* is not bound by any element inside the sentence. In such a case, the reflexive is usually bound pragmatically by the speaker. In Fig. 4 (b), on the other hand, the verb phrase designated by an arrow dominates an occurrence of *zibun* and an instance of p[*sbj*] is included in its **subcat** value. Thus, *zibun* is bound by this p[*sbj*], unifying the p[*sbj*] in the value of **subcat** and the p[*sbj*] in the value of **refl**. The p[*sbj*] is further


v[**subcat** ϕ, **refl** {p[*sbj*]}]
p[*sbj*]
v[**subcat** {p[*sbj*]}, **refl** {p[*sbj*]}]
Ken-ga
p[*obj*, **refl** {p[*sbj*]}]
v[**subcat** {p[*sbj*],p[*obj*]}]
zibun-wo
aisitei

(a) Binding of *Zibun* by the Speaker


v[**subcat** ϕ, **refl** ϕ]
p[*sbj*]
⇒v[**subcat** {p[*sbj*]}, **refl** ϕ]
Ken-ga
p[*obj*, **refl** {p[*sbj*]}]
v[**subcat** {p[*sbj*],p[*obj*]}]
zibun-wo
aisitei

(b) Binding of *Zibun* by *Ken*

Figure 4: An Application of Binding Feature Principle to **refl**

unified with the subject of the sentence *Ken-ga* in accordance with the Subcat Feature Principle and the intended interpretation that *zibun* refers to Ken is obtained.

4 Principles of JPSG Parser

In this section, we will describe the JPSG parser currently implemented at ICOT (Institute for New Generation Computer Technology—Institute for Japan's 'Fifth Generation' Computer Project). As described in the previous section, compared with the traditional context free grammar descriptions that state a separate phrase structure rule corresponding to each local phrase structure, the phrase structure rule in JPSG is stated in a more general, abstract form. This has been made possible by considering grammatical categories as sets of features, and by stating generalizations that are found in the grammar in the form of grammatical principles that are independent of phrase structure rules.

The process of parsing a sentence can be construed as a process where the internal structure of the input sentence is made precise and concrete through the combination of a few grammatical rules and several general syntactic principles. In other words, this is a process of determining the values of features starting with the information given by each lexical item in a phrase structure tree. This process is generally called *unification*. Note, however, that the use of the term *unification* is slightly different from that in Prolog; it is extended in such a way that will facilitate linguistic descriptions and has been called *constraint unification* ('conditioned unification' in earlier literature) [6,7]. We are currently experimenting on a parser based on this extended notion of unification. In the following, we will describe the notion of constraint unification and its application to parser implementation. The actual working system is implemented on a logic programming language, which is an extension of Prolog.

4.1 Constraint Unification

Grammatical features consist of feature names and their values. Unification of features thus entails the unification of feature values corresponding to the same feature names. But these feature values are often only partially specified; for example, they are only known to fall within a certain range. Moreover, they may be constrained to stand in certain relationships with other feature values. Thus, unification of patterns as employed in the ordinary Prolog implementation is insufficient to process this kind of constraint.

One familiar way to cope with this kind of situation is the use of the so-called 'procedure attachment'. However, there are certain obvious defects in this approach. First, it is generally not known when to evaluate these attached procedures, which should properly constrain the values of a given variable. This means that evaluations of attached procedures might occur when they are *not* required, on the one hand, and might not occur when they *are* required, on the other. Secondly, procedure attachment is possible only when the direction of the flow of information is predetermined. If grammatical descriptions are to be used both in parsing and production of sentences, which we aim at, this will pose a serious problem.

These considerations led us to employ *constraint unification.* In constraint unification, we unify two patterns, both of which have accompanying constraints that are imposed on the variables appearing in them. These patterns are called *constrained patterns.* In performing unification, it is first decided whether a given set of constrained patterns are unifiable in such a way to satisfy the accompanying constraints. When they are known to be unifiable, the unified pattern is produced together with a new set of constraints that conform with the previous constraints. The resulting constrained pattern will represent the set that can be obtained as the intersection of the sets represented by the given constrained patterns.

In order for constraint unification to be possible, the constraints accompanying the patterns must be *modular* in the following sense [6]:

(4.1) *Modularity*
The constraints accompanying the constrained patterns must be sets of atomic expressions whose arguments are all variables. For each variable in the pattern, at most one constraint can be allowed. That is, each variable can appear only once in a constraint.

This restriction assures the consistency among the constraints. At the same time, it effectively eliminates any redundancy in the constraints. Since inconsistency and redundancy is rare in the description of any natural system, it does not seem to pose any difficulty for JPSG, or the description of natural language for that matter. Constraint unification can be thought of as a process of information accumulation with respect to the patterns involved. Borrowing the Prolog notation, constraint unification can be expressed as a predicate of the following form:

(4.2) unify(Pat1, Pat2, Constraint, NewConstraint)

where 'Pat1' and 'Pat2' represent the patterns to be unified, 'Constraint' a list consisting of all the constraints accompanying the patterns, and 'NewConstraint' the resulting constraint. For example, suppose that predicates designated by c1, c2, and c3 are defined as follows:[4]

```
(4.3) c1(a, b).
      c1(c, U).
      c2(a, U).
      c2(b, a).
      c3(U, [U, a, V]).
      c3(a, [b, U, V]) :- c1(U, V).
```

In this example,

```
(4.4) unify([A, [B, a, C], D], [X, Y, Z], [c1(A, B), c2(C, D), c3(X, Y)], NewC)
```

succeeds, with the following resultative constraint.

```
(4.5) NewC = c4(A, B, C, D).
```

where

```
(4.6) c4(a, b, b, a).
      c4(c, c, U, V :- c2(U,V).
```

Even though the predicate 'unify' could be simulated in Prolog, we are currently developing an extended Prolog system which includes the predicate 'unify' as a built-in predicate.

4.2 An example of Parser Implementation

4.2.1 Lexical Entries

All lexical entries are represented as constrained patterns of the following form:

```
(4.7) lex(Entry, cat(head(Pos, Pform, Gr, Pas), Subcat, Sem, Slash, Refl), Const).
```

where 'cat' is a dummy predicate whose arguments consist of all the feature specifications of the category involved (head features, **subcat**, **sem**, **slash**, and **refl**). 'Const' corresponds to the list of constraints on the variables involved in these feature specifications.

4.2.2 Parsing Algorithm

Constraint unification is implemented independently of the specific parsing algorithm involved. That is, it can be used with any available parsing strategy. Here, as an example of how this can be utilized, we will present below a top-down parsing algorithm which is reminiscent of DCG [9].

The phrase structure rule for complementation can be expressed as follows (**sem**, **slash**, and **refl** are suppressed for brevity's sake):

[4]As with Prolog, capital letters designate variables, while lower-case letters constants.

(4.8) *Top-Down Parsing Algorithm for Complementation*

```
rule(cat(Head,RestSubcat),NewConsts,First,Last) :-
    rule(Cat_of_Complement,Constraint_of_Complement,First,Middle),
    rule(cat(Head,[Complement|RestSubcat]),Constraint_of_Head,Middle,Last),
    append(Constraint_of_Head,Constraint_of_Complement,Consts),
    unify(Cat_of_Complement,Complement,Consts,NewConsts).
```

where 'append' is the usual Prolog predicate, whose third argument is the concatenated list of its first and second argument lists. The above DCG-like rule incorporates both the Head Feature Principle and the Subcat Feature Principle and can be used as part of a top-down parser.

5 Concluding remarks

We have described the general framework of JPSG, a phrase structure grammar for Japanese, and its principal implementation mechanisms. Even though JPSG is considered to be among the recent grammatical theories based on unification, such as GPSG and HPSG, many of the concepts in the grammatical framework are extended to facilitate adequate description of natural language in general and Japanese in particular. In this paper, we have focused on our extension of the subcategorization feature, on the one hand, and of unification, on the other. We have thus far completed the basic design of core parts of the parser, and intend to develop a full system and the related tools on a sequential inference machine PSI [11] at ICOT.

Acknowledgments The study reported in this article is the result of discussions among our colleagues in the JPSG working group at ICOT. We are grateful to both the current and the previous members of this project.

References

[1] Barwise, J. and Perry, J. : *Situations and Attitudes*, MIT Press, 1983.

[2] Gazdar, G., Klein, E., Pullum, G.K., and Sag, I.A. : *Generalized Phrase Structure Grammar*, Basil Blackwell, 1985.

[3] Gunji, T. : Phrase Structure Grammars, *J. Inf. Proc. Soc. Japan*, Vol. 27, No. 8 (1986), pp. 868–875 (in Japanese).

[4] Gunji, T. : Subcategorization and Word Order, in Nagao, M. (ed.) *Language and Artificial Intelligence: Proceedings of an International Symposium on Language and Artificial Intelligence held in Kyoto, Japan, 16–21 March, 1986*, North-Holland, 1987, pp. 51–73.

[5] Gunji, T. : *Japanese Phrase Structure Grammar: A Unification-based Approach*, D. Reidel, 1987.

[6] Hasida, K. : Conditioned Unification for Natural Language Processing, *Proc. 11th Int. Conf. on Comput. Linguistics*, (1986), pp. 85–87.

[7] Hasida, K. and Sirai, H. : Conditioned Unification, *Comput. Softw.*, Vol. 3, No. 4 (1986), pp. 28–38 (in Japanese).

[8] Montague, R. : Formal Philosophy, R.H. Thomason (ed.), Yale University Press, 1974.

[9] Pereira, F. and Warren, D. : Definite Clause Grammar for Language Analysis: A Survey of the Formalism and a Comparison with Augmented Transition Networks, *Artif. Intell.*, Vol. 13, (1983), pp. 231-278.

[10] Pollard, C.J. and Sag, I.A. : *Information-Based Syntax and Semantics, Vol. 1: Fundamentals*, CSLI Lecture Notes No. 12, Center for the Study of Language and Information, Stanford University, 1988.

[11] Yokota, M., Yamamoto, A., Taki, K., Nishikawa, H. and Uchida, S. : The Design and Implementation of a Personal Sequential Inference Machine (PSI), *ICOT TR-045*, Institute for New-Generation Computer Technology, 1984.