On Island Constraints—A Phrase Structure Grammar Perspective

HARADA Yasunari
Waseda University

In this article we will consider how the so-called "island constraints" are to be incorporated into a phrase structure grammar description of (a fragment of) English Grammar.

"Island Constraints" have long been the central issue of generative transformational studies of English syntax. Although researchers working within unification-based grammars do not necessarily share this interest, we have to consider what a possible 'solution' to this problem within a phrase structure grammar approach to syntax might look like and how it might affect our understanding of the 'adequacy' of our research objective. Thus we will define a very small fragment of English, outline how unbounded dependencies are to be dealt with in that fragment and then go on to discuss how 'island constraints' are to be guaranteed.

0 Constraint-Unification-Based Phrase Structure Grammar

In a constraint-unification-based phrase structure grammar, syntactic regularities are to be stated in terms of local phrase structures and constraints obtaining among feature specifications involved in a given local phrase structure. Here a description of a fragment of a natural language such as English or Japanese is determined if and only if we (1) define what objects constitute grammatical categories in that language, (2) state what phrase structure rules are to be utilized in that language and (3) specify what lexical items belong to what syntactic category or categories. As with other recent grammatical theories, we conceive of grammatical categories as bundles of feature specifications rather than non-decomposable monadic objects. Also, we permit partial specifications of these categories. Thus, phrase structure rules in our fragment will refer to grammatical categories with very few feature specifications, which enables us to efficiently state syntactic regularities.

When embedded into an environment where constraint-unification is executed, our grammar will sanction only the grammatical configurations of (the fragment of) English defined. Partially specified categories that are mentioned in phrase structure rules are 'unified' or matched with more richly specified categories that are assigned to lexical items. In this way, our grammar determines (1) whether a given string of words constitute a grammatical sentence (or some other category) of (our fragment of) English, (2) what are possible strings in the language defined, and (3) what parse trees are to be assigned to these strings.

0.1 Categories and features

As an example of how our description of English might look like, we will give you some typical features together with their intuitive or heuristic 'meanings' and possible ranges of their values. This is of course a very limited subset of features that are needed to adequately describe the grammar of English.

(0.1.1)	pos	part of speech	{n, v, p, a, det}
	pn	person and number	{nil, 1s, 2s, 3s, 1p, 2p, 3p}
	case	case	{nil, nom, poss, acc}
	form	verb form	{nil, base, fin, presp, pastp}
		preposition	{*of, at, on, in, for, to*}
	spec	specifier	list of categories
	subcat	complement	list of categories
	comp	complementizer	list of variables
	sem	semantics	some expression
	gap	syntactic gap	list of categories
	bind	variable	list of variables

In what follows, categories are designated by a left square bracket ("[") followed by an indefinite number of feature specifications separated by commas (",") followed by a right square bracket ("]"). A feature specification is maximally a feature name followed by its value. However, when the value uniquely determines the name, the name can be omitted. Also, when the value is nil or a null-list (< >) the entire feature specification can be omitted. Finally a category of the form [pos *P*, ..., sem *S*] is sometimes designated as *P*[...]:*S*. Some examples follow, together with symbols often used in traditional generative literature. Note that traditional symbols do not necessarily bear as much information as category designations defined here.

(0.1.2) NP [pos n, subcat < >, spec < >, sem **m'**]
n[ ]:**m'**
PP [pos p, subcat < >, spec < >, form *for*]
p[*for*]
VP [pos v, spec <[pos n, subcat nil, sem X]>, form fin,
sem **love'**(X,**j'**)]
v[fin, spec <n[ ]:X>]:**love'**(X,**j'**)
P [pos p, subcat <[pos n, subcat < >, case acc, sem **j'**]>, form on]
p[subcat <n[acc]:**j'**>, on]
V [pos v, subcat <[pos n, subcat < >]>,
spec<[pos n, subcat < >, pn 3s]>]
v[subcat <n[ ]>, spec<n[3s]>]

0.2 Phrase Structure Rules and Feature Inheritances

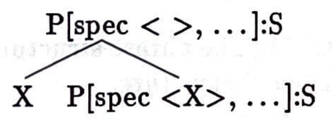
Partial specification of grammatical categories enables us to efficiently encapsulate a great part of English syntax into the following two phrase structure rules.

(0.2.1) a. specification

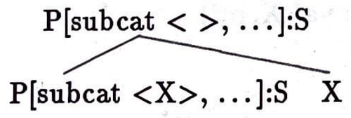
```
    Mr → Sp Hd
                 where spec @ Hd = <Sp|spec @ Mr>,
                       subcat @ Mr = subcat @ Hd = nil,
                       sem @ Mr = sem @ Hd,
                       head @ Mr = head @ Hd

b. complementation

    Mr → Hd Ct
                 where subcat @ Hd = <Ct|subcat @ Mr>,
                       spec @ Mr = spec @ Hd,
                       sem @ Mr = sem @ Hd,
                       head @ Mr = head @ Hd
```

Typically these two phrase structure rules sanction the following local phrase structures, respectively. Note that in English, the length of the value of spec cannot be more than 1, but the same does not hold for the length of the value of subcat. However, since an adequate exposition of how double object constructions and control phenomena are to be handled in the framework we have in mind here would take us too far afield, we will restrict our attention to cases where the length of the value of subcat is less than 2.

(0.2.2) a.



b.



The constraints or conditions that follow "where" (0.2.1.a,b) are the clauses in which syntactic regularities are stated. These conditions are called feature inheritances. Here an expression of the form "Feature_name @ Category_designator" refers to the value of the feature designated by Feature_name with respect to the grammatical category referred to by Category_designator. The symbol "=" represents unification. However, unification here employed disregards the values of bind. The expression "head @ C1 = head @ C2" is a short hand way of repeating expressions of the form "F @ C1 = F @ C2" with F ranging over head_features, where head_features = {pos, pn, case, form}. Here and in what follows, constants are designated by names beginning with a lowercase letter and variables and meta-variables are designated by names beginning with an uppercase letter.

Let us take a very simple sentence such as "John loves Mary" and see how these phrase structure rules are involved in admitting that sentences like this are grammatical in English.

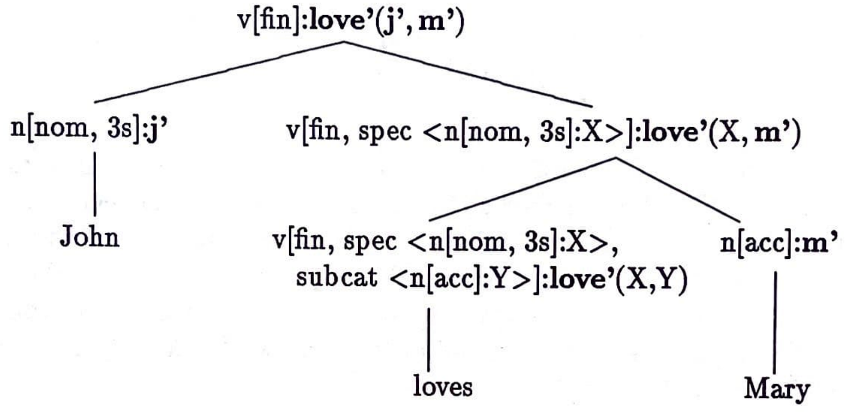
First of all we have to remember that in a phrase structure grammar account of English syntax, all lexical items or words are supposed to be given rich syntactic information in the lexicon, where syntactic and semantic information concerning each and every word is stored. For instance, a transitive verb "love" will be specified in the lexicon as shown in (0.2.3).

(0.2.3) loves ⊨ v[fin, spec <n[nom, 3s]:X>, subcat <n[acc]:Y>]:**love'**(X,Y)

An expression of the form "Spelling ⊨ Category" asserts that a lexical entry whose representation is given as Spelling is assigned feature specifications designated by Category.

Our two phrase structure rules assigns the following parse tree to the string "John loves Mary".

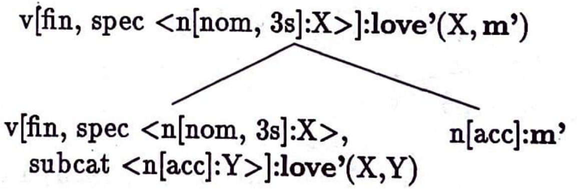
(0.2.4) John loves Mary.



v[fin]:love'(j', m')
n[nom, 3s]:j'
v[fin, spec <n[nom, 3s]:X>]:love'(X, m')
John
v[fin, spec <n[nom, 3s]:X>, subcat <n[acc]:Y>]:love'(X,Y)
n[acc]:m'
loves
Mary

Let's take a look at the local structure in (0.2.5). The phrase structure rule given in (0.2.1.b), or complementation, sanctions this local structure.

(0.2.5)

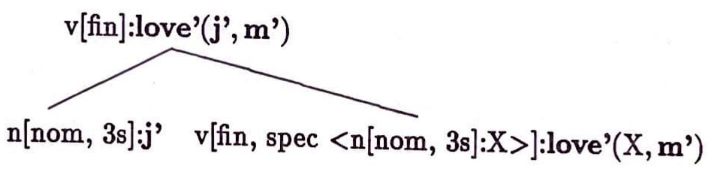


v[fin, spec <n[nom, 3s]:X>]:love'(X, m')
v[fin, spec <n[nom, 3s]:X>, subcat <n[acc]:Y>]:love'(X,Y)
n[acc]:m'

If we match the three grammatical categories in (0.2.5) against those in (0.2.1.b), more specifically, v[fin, spec <n[nom, 3s]:X>]:**love'**(X, **m'**) against Mr, v[fin, spec <n[nom, 3s]:X>, subcat <n[acc]:Y>]:**love'**(X,Y) against Hd, and n[acc]:**m'** against Ct, we see that all the conditions that follow "where" are satisfied. Therefore, the three categories in (0.2.5) are said to "unify" with the three categories in (0.2.1.b) and complementation is said to "sanction" this local phrase structure. Incidentally, variable Y in the semantic representation of the transitive verb "love" is bound to or unified with the constant **m'**, which is the semantic representation associated with the proper noun "Mary", as a side-effect of this unification.

Likewise, the 'upper part' of the parse tree in (0.2.4), namely the local structure in (0.2.6) is sanctioned by specification in (0.2.1.a).

(0.2.6)

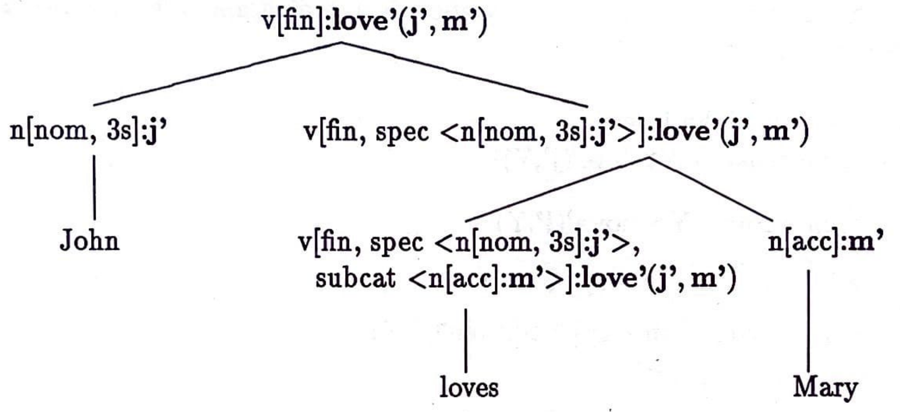


Here, if we match v[fin]:**love'(j', m')** against Mr, n[nom, 3s]:**j'** against Sp, and v[fin, spec <n[nom, 3s]:X>]:**love'(X, m')** against Hd, all constraints are satisfied. As above, variable X in the semantic representation of the verb phrase is bound to **j'**, which is the semantic representation of the subject proper noun. The reason a prepositional phrase cannot be the subject of a transitive verb like "loves" is that its value for pos would be p, thus preventing it from unifying with Sp in the local structure involved. Also, a plural noun phrase cannot be the subject here, because the value of pn with respect to Sp is specified as 3s, rather than 3p.

We have reasons to believe that except for constructions that involve dislocation of elements, only four phrase structure rules, namely complementation, specification, adjunction and coordination are responsible for almost all English sentence constructions. However, we have very little, if any, to say about adjunction or coordination in this article.

Strictly speaking, our constraint-unification executive would not produce parse-trees as shown in (0.2.4) but rather something like (0.2.7).

(0.2.7) John loves Mary.


v[fin]:love'(j', m')
n[nom, 3s]:j'
v[fin, spec <n[nom, 3s]:j'>]:love'(j', m')
John
v[fin, spec <n[nom, 3s]:j'>, subcat <n[acc]:m'>]:love'(j', m')
n[acc]:m'
loves
Mary

That is, once unification binds a variable to a constant, all relevant occurrences of the same variable is bound to that constant throughout a given representation. However, if we give parse trees like (0.2.7), the binding process is quite obscured to readers unfamiliar with our approach presupposed here. Therefore, we will talk as if parsing is processed bottom-up, and give parse trees the way (0.2.4) is given, rather than the way (0.2.7) is given.

For brevity, we will omit reference to features such as pn or case in what follows.

1 Unbounded Dependencies in Phrase Structure Grammar

Unbounded dependencies are guaranteed through chains of local constraints in the phrase structure analysis of English we are here considering. The feature gap plays a central role in this, binding syntactically and semantically the dislocated grammatical element and the syntactic gap it binds in an unbounded dependency construction.

Typical examples of unbounded dependencies are sentences such as those in (1.0.1). Here, however, in order not to complicate our grammar to something overly loaded with new lexical items or phrase structure rules that are responsible for Subject Auxiliary Inversion constructions, let us take sentences in (1.0.2). Sentences in (b) and (d) are added because in our grammar, subjects and objects are treated quite differently.

(1.0.2) a. what did you see _

b. what do you think that John saw _

(1.0.3) a. [I wonder] who John loves

b. [I wonder] who loves John

c. [I wonder] who Mary thinks John loves

d. [I wonder] who Mary thinks loves John

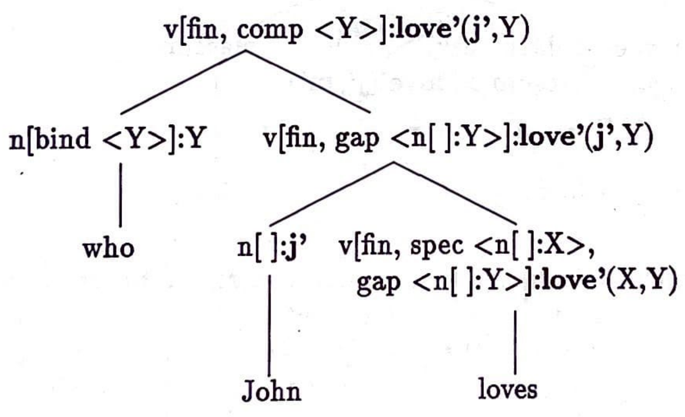
Here, we are presupposing a lexical entry of the following form.

(1.0.4) thinks ⊨ v[fin, spec<n[ ]:X>, subcat <v[fin]:Y>]:**think'**(X,Y)

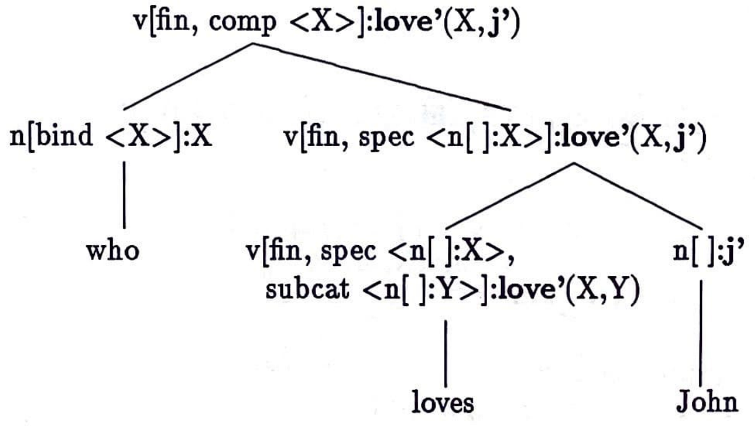
For the (embedded) sentences in (1.0.2) our grammar will assign the following parse trees. Explanations of what these symbols in the diagrams are supposed to mean will follow shortly.

(1.0.4) [I wonder] who John loves

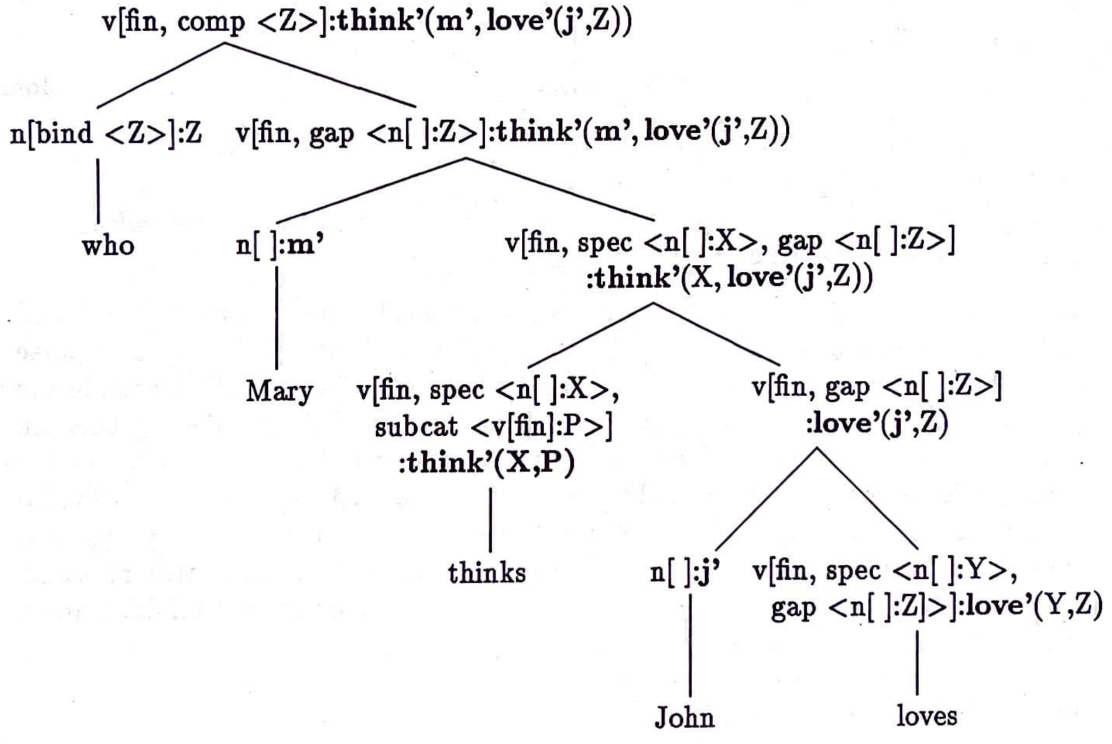
wonder'(speaker,<Y>,love'(j',Y))


v[fin, comp <Y>]:love'(j',Y)
n[bind <Y>]:Y
v[fin, gap <n[ ]:Y>]:love'(j',Y)
who
n[ ]:j'
v[fin, spec <n[ ]:X>, gap <n[ ]:Y>]:love'(X,Y)
John
loves

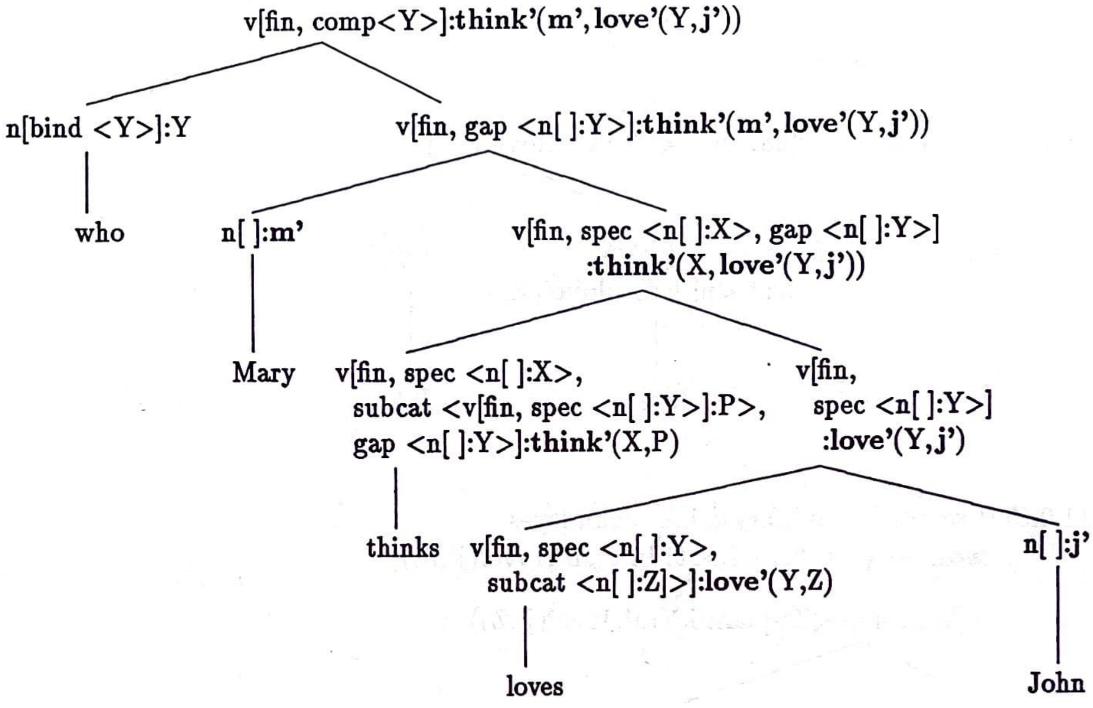
(1.0.5) [I wonder] who loves John
wonder'(speaker,<X>, **love'**(X,j'))


v[fin, comp <X>]:love'(X,j')
n[bind <X>]:X
v[fin, spec <n[ ]:X>]:love'(X,j')
who
v[fin, spec <n[ ]:X>, subcat <n[ ]:Y>]:love'(X,Y)
n[ ]:j'
loves
John

(1.0.6) [I wonder] who Mary thinks John loves
wonder'(speaker,<Z>, **think'**(m', **love'**(j',Z)))


v[fin, comp <Z>]:think'(m', love'(j',Z))
n[bind <Z>]:Z
v[fin, gap <n[ ]:Z>]:think'(m', love'(j',Z))
who
n[ ]:m'
v[fin, spec <n[ ]:X>, gap <n[ ]:Z>] :think'(X, love'(j',Z))
Mary
v[fin, spec <n[ ]:X>, subcat <v[fin]:P>] :think'(X,P)
v[fin, gap <n[ ]:Z>] :love'(j',Z)
thinks
n[ ]:j'
v[fin, spec <n[ ]:Y>, gap <n[ ]:Z]>]:love'(Y,Z)
John
loves

(1.0.7) [I wonder] who Mary thinks loves John
wonder'(speaker,<Y>,**think'**(m',**love'**(Y,j')))


v[fin, comp<Y>]:think'(m',love'(Y,j'))
n[bind <Y>]:Y
v[fin, gap <n[ ]:Y>]:think'(m',love'(Y,j'))
who
n[ ]:m'
v[fin, spec <n[ ]:X>, gap <n[ ]:Y>] :think'(X,love'(Y,j'))
Mary
v[fin, spec <n[ ]:X>, subcat <v[fin, spec <n[ ]:Y>]:P>, gap <n[ ]:Y>]:think'(X,P)
v[fin, spec <n[ ]:Y>] :love'(Y,j')
thinks
v[fin, spec <n[ ]:Y>, subcat <n[ ]:Z]>]:love'(Y,Z)
n[ ]:j'
loves
John

1.1 Binding Features

The feature gap takes a list consisting of grammatical categories as its value and propagates information concerning the existence of syntactic gaps through a parse tree. This feature is equivalent, for the most part, to the feature called slash in the previous GPSG and/or HPSG literature. Note incidentally, that binding between semantic representations of dislocated elements and variables that appear in the semantic representations of gaps they bind is also achieved as 'side-effects' of unification chains. The feature bind takes a list of variables as its value and propagates information concerning semantic variables of relative and interrogative constructions. This feature is somewhat similar to the feature wh in previous GPSG studies. These two are called binding features.

(1.1.1) binding_features = {gap, bind}

Binding feature inheritance states local constraints obtaining among values of binding features with respect to grammatical categories involved in a given local phrase structure. In general, the following condition holds in all local structures where binding features are not bound.

(1.1.2) binding feature inheritance

Mr $\rightarrow$ C1 C2

where F @ Mr = F @ C1 + F @ C2
if F ∈ binding_features

Here "+" designates a list operation not unlike concatenation. For instance, < > = < > + < >, <X> = <X> + < >, <X> = < > + <X> and so on. However, non-distinct elements are merged rather than repeated. Thus, <X> = <X> + <X>, <X|Y> = <X> + <X|Y> and so on. Typically (1.1.2) sanctions local structures of the following forms, among others.

(1.1.3) a.

$$\begin{tikzcd}[column sep=small] & {[\text{gap } \langle X \rangle]} \arrow[dl, no head] \arrow[dr, no head] & \\ {[\text{gap } \langle X \rangle]} & & {[\text{gap } \langle\, \rangle]} \end{tikzcd}$$

b.

$$\begin{tikzcd}[column sep=small] & {[\text{gap } \langle X \rangle]} \arrow[dl, no head] \arrow[dr, no head] & \\ {[\text{gap } \langle\, \rangle]} & & {[\text{gap } \langle X \rangle]} \end{tikzcd}$$

c.

$$\begin{tikzcd}[column sep=small] & {[\text{bind } \langle X \rangle]} \arrow[dl, no head] \arrow[dr, no head] & \\ {[\text{bind } \langle X \rangle]} & & {[\text{bind } \langle\, \rangle]} \end{tikzcd}$$

d.

$$\begin{tikzcd}[column sep=small] & {[\text{bind } \langle X \rangle]} \arrow[dl, no head] \arrow[dr, no head] & \\ {[\text{bind } \langle\, \rangle]} & & {[\text{bind } \langle X \rangle]} \end{tikzcd}$$

Earlier we mentioned that our unification disregards the values of bind. In fact, we have reasons to believe that something like (1.1.4) really captures how bind and gap are inherited with respect to complentation and specification. Consideration of how constructions involving double wh questions and parasitic gaps along with violin-sonata sentences are to be admitted in our framework would lead to the these formulations. However, an adequate explanation of these would justify another article. Thus, we assume simpler inheritance conditions as given in (1.1.2) in the discussions that follow.

(1.1.4) an alternative formulations of phrase structure rules
(Here "⊕" designates concatenation.)

a. complementation

$$\begin{tikzcd}[column sep=small] & {\text{H[subcat L1, gap L2}\oplus\text{L3, bind L4}\oplus\text{L5]}} \arrow[dl, no head] \arrow[dr, no head] & \\ {\text{H[subcat } \langle \text{C[gap L6]} \mid \text{L1} \rangle\text{, gap L2, bind L4]}} & & {\text{C[gap L6}\oplus\text{L3, bind L5]}} \end{tikzcd}$$

b. specification

$$\begin{tikzcd}[column sep=small] & {\text{H[gap L1}\oplus\text{L2, bind L4}\oplus\text{L5, comp L6}\oplus\text{L7]}} \arrow[dl, no head] \arrow[dr, no head] & \\ {\text{C[gap L1, bind L4}\oplus\text{L6]}} & & {\text{H[spec } \langle \text{C} \rangle\text{, gap L1}\oplus\text{L2, bind L5}\oplus\text{L7]}} \end{tikzcd}$$

1.2 Lexical Rules

Lexical rules assert regularities that hold among lexical entries. To take a trivial example, given an infinitival form of a verb, we can 'predict' rather safely that a third person singular form of that verb exists, whose spelling is obtained by adding an "s" at the end of the spelling of the infinitival form with some orthographical modifications. Thus, something like (1.2.1) will be part of the lexicon (and grammar) of English.

(1.2.1) third person singular lexical rule
Vs ⊨ [fin, spec <n[nom, 3s]>] if V ⊨ [inf, spec <n[ ]>]
where Vs tpsf V

Here, "if" is a relation obtaining among two lexical entries. An expression of the form given in (1.2.1) states that a lexical entry that appear on the left-hand side of "if" exists in the lexicon if a lexical entry that appear on the right-hand side of "if" exists in it. All feature specifications including spellings and semantic representations that are not explicitly stated in the rule are the same between the two lexical entries. In the example given in (1.2.1) "tpsf" is to be construed as a binary relation that holds between an infinitival form representation of a verb and its third person singular form representation.

Some lexical rules play an important role in our analysis of unbounded dependency constructions. For instance, (1.2.2) is indispensable.

(1.2.2) gap introduction lexical rule
v[subcat < >, gap <n[ ]>] if v[subcat <n[ ]>, gap < >]

In the fragment defined in the previous section, "loves" is specified as a transitive verb, occurring in environments immediately preceding a noun phrase. However, as classical arguments show, they appear without apparent object noun phrases in an unbounded dependency constructions. Thus, only (b) is ungrammatical in (1.2.3).

(1.2.3) a. John loves Mary.
b. *John loves.
c. Mary, John loves.
d. Jane, Mary thinks John loves.

Given a lexical rule of the form in (1.2.2) and a lexical entry of the form in (0.2.3) we obtain the following lexical entry, which is involved in the sentences (1.2.3.c,d).

(0.2.3) loves ⊨ v[fin, spec <n[nom, 3s]:X>, subcat <n[acc]:Y>]:**love'**(X,Y)
(1.2.4) loves ⊨ v[fin, spec <n[nom, 3s]:X>, gap <n[ ]:Y>]:**love'**(X,Y)

Note that the lexical rule in (1.2.2) explicitly refers to subcat. Thus, (1.2.5) is not part of English lexicon.

(1.2.5) (this is not part of English lexicon)
loves ⊨ v[fin, gap <n[ ]:X>, subcat <n[ ]:Y>]:**love'**(X,Y)

Then what about sentences like (1.0.2.d).

(1.0.2.d) [I wonder] who Mary thinks loves John

Here, following the analysis of Gazdar (1981) we assume the following lexical rule.

(1.2.6) subject extraction lexical rule
    v[subcat <v[fin, spec <n[ ]:X>]>, gap <n[ ]:X>]
    if v[subcat <v[fin, spec < >, gap < >]>]

Since we have a lexical entry of the form in (1.0.3), a lexical entry of the form in (1.2.7) is obtained if we assume that "thinks" is a legitimate input to this rule.

(1.0.3) thinks ⊨ v[fin, spec<n[ ]:X>, subcat <v[fin]:Y>]:**think'**(X,Y)

(1.2.7) thinks ⊨ v[fin, spec<n[ ]:X>, subcat <v[fin, spec <n[ ]:Y>]:Z>,
    gap <n[ ]:Y>]:**think'**(X,Z)

1.3 Binding

We first saw how information regarding the existence of gaps are to be propagated, and then introduced lexical rules that introduce gaps. (The bottom- up expressions are intended only as a means of facilitating the understanding of the readers.) What remains to be discussed is how binding is achieved.

We have to take into consideration two kinds of binding. Topicalization is syntactic binding of syntactic gaps, whereas complementization is binding of semantic variables.

1.3.1 Syntactic Binding

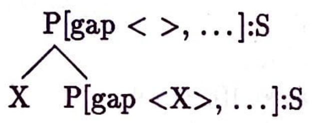
Information concerning the existence of syntactic gaps has to be somehow resolved or bound in some local phrase structure. The following phrase structure rule sanctions local structures where this binding takes place.

(1.3.1.1) topicalization
    Mr → Bdr Hd
        where gap @ Hd = <Bdr|gap @ Mr>,
            subcat @ Mr = subcat @ Hd = nil,
            spec @ Mr = spec @ Hd = nil,
            sem @ Mr = sem @ Hd,
            head @ Mr = head @ Hd

In this local structure, binding occurs with respect to gap @ Bee. Therefore, binding feature inheritance does not hold in this local structure with respect to gap @ Bee and gap @ Mr. The relation among them are explicitly stated as a constraint.

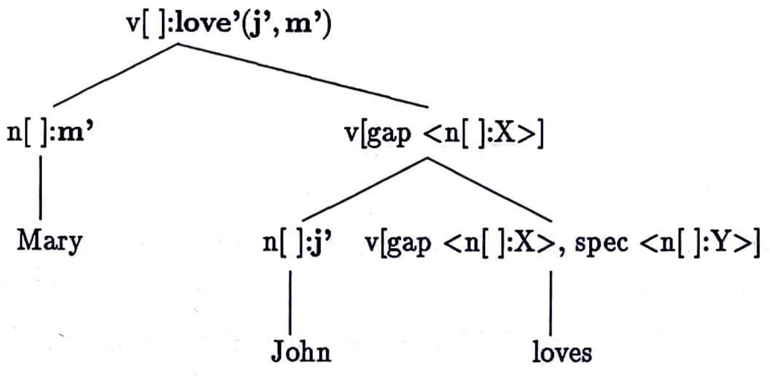
Graphically, the following structure will show what topicalization is all about.

(1.3.1.2)



This will sanction sentences like (1.3.1.3).

(1.3.1.3) Mary, John loves



1.3.2 Semantic Binding

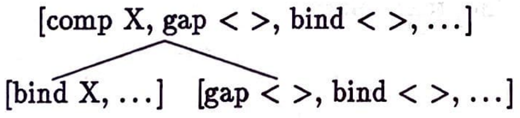
Besides syntactic binding of gaps, we have to take into consideration semantic binding of variables that occur in a relative or interrogative constructions. We call this complementization.

```
(1.3.2.1) complementization
          Mr → C1 C2
                      where comp @ Mr = bind @ C1,
                            gap @ Mr = nil
```

Here, since binding occurs with respect to bind @ C1, binding feature inheritance does not hold between bind @ C1 and bind @ Mr.

Graphically, (1.3.2.2) will tell you what this is all about.

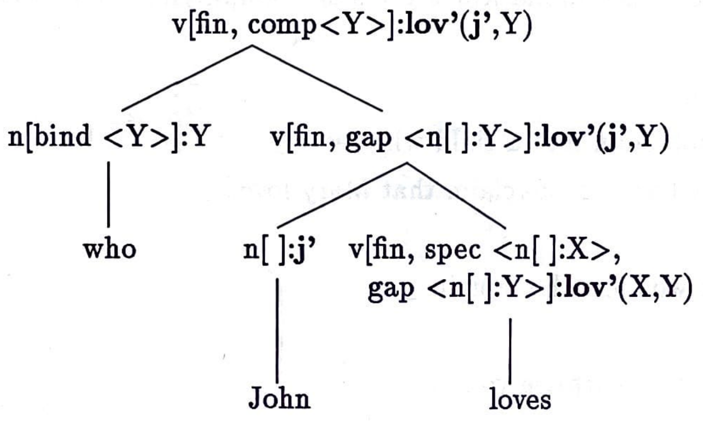
(1.3.2.2)



Although we give this as a phrase structure rule, or constraints obtaining among feature specifications involved in a local phrase structure, it can never 'stand alone'. Complementization concurs with topicalization and specification. Or, we can think of complementization as unifying with other phrase structure rules.

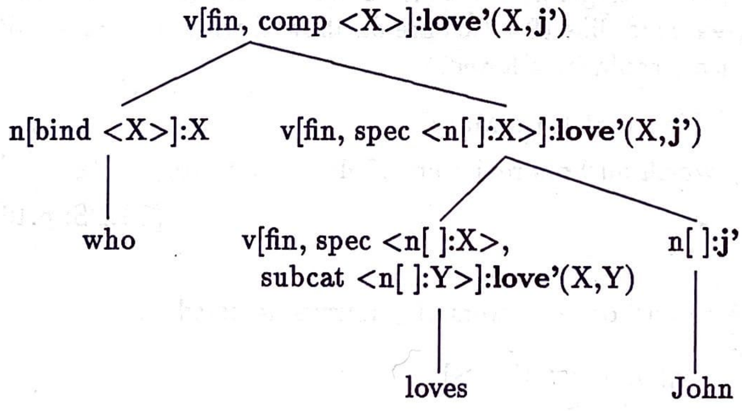
```
(1.3.2.3) complementization unified with topicalization
          Mr → Bdr Hd
                      where gap @ Mr = nil,
                            gap @ Hd = <Bdr>,
                            comp @ Mr = bind @ Bdr,
                            sem @ Mr = sem @ Hd,
                            head @ Mr = head @ Hd,
                            subcat @ Mr = subcat @ Hd = nil,
                            spec @ Mr = spec @ Hd = nil
```

(1.3.2.4) [I wonder] who John loves


v[fin, comp<Y>]:**lov'**(j',Y)
n[bind <Y>]:Y
v[fin, gap <n[ ]:Y>]:**lov'**(j',Y)
who
n[ ]:j'
v[fin, spec <n[ ]:X>, gap <n[ ]:Y>]:**lov'**(X,Y)
John
loves

(1.3.2.5) complementization unified with specification
Mr → Sp Hd
where spec @ Hd = <Sp|spec @ Mr>,
comp @ Mr = bind @ Hd,
gap @ Mr = gap @ Hd = nil,
sem @ Mr = sem @ Hd,
head @ Mr = head @ Hd,
subcat @ Mr = subcat @ Hd = nil

(1.3.2.6) [I wonder] who loves John


v[fin, comp <X>]:**love'**(X,j')
n[bind <X>]:X
v[fin, spec <n[ ]:X>]:**love'**(X,j')
who
v[fin, spec <n[ ]:X>, subcat <n[ ]:Y>]:**love'**(X,Y)
n[ ]:j'
loves
John

2 Island Constraints

In this section we will consider how the so-called "island constraints" phenomena are to be guaranteed in a phrase structure grammar description of English.

2.1 Examples

What follows are some typical ungrammatical sentences exemplifying island constraint violations.

(2.1.1) complex NP constraint

a. *[I wonder] which book John met a child who read _

b. *[I wonder] who John believes the claim that Mary loved _

wh-island

c. *which book did you wonder who bought _

subject condition

d. *who did a story about _ surprise you

The ungrammatical examples (2.1.1.a) and (2.1.1.c) both violate the condition that gap @ Mr = nil in local structures where complementization is to take place. We will discuss this point shortly.

It is difficult to account for the ungrammaticality of (2.1.1.b) from our point of view, because with respect to local structures involved, it is quite similar to the sentence in (2.1.2), which is grammatical.

(2.1.2) [I wonder] who John believes that Mary loves _

This disparity could be explained from a cognitive point of view. See Hasida's article in this volume for detail.

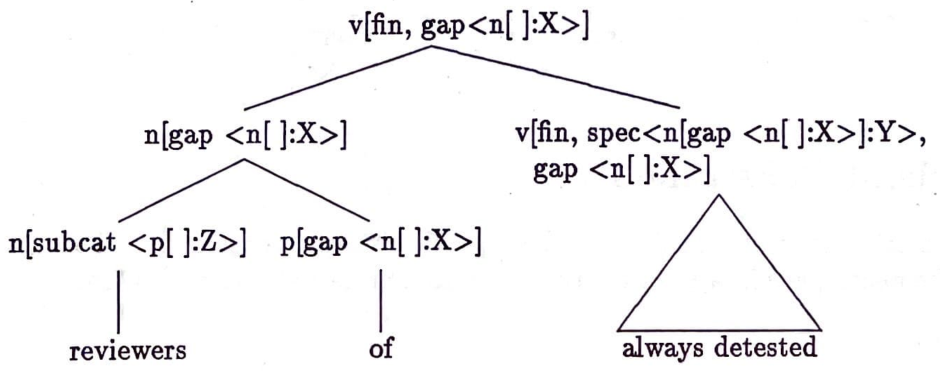
Our simpler grammar with binding feature inheritance as stated in (1.1.2) would admit those strings as given in (2.1.1.d), although sentences like (2.1.3.a) would be disallowed simply because our gap inheritance lexical rule specifically refers to subcat rather than spec in the input category. However, if we assume phrase structure rules as given in (1.1.4), sentences like (2.1.1.d) are disallowed while sentences with parasitic gaps within subjects could be allowed.

(2.1.3) a. *Who did John think that loves Mary?

b. Kim wondered which authors reviewers of always detested.

[GKPS: p.163]

(2.1.4) Kim wondered which authors reviewers of _ always detested _.



A reasonable explanation of how this is achieved will take us too far afield, so let's suffice it to say that in the 'top' local phrase structure in (2.1.4) where specification takes place, the relations in (2.1.5.a) hold among values for gap with respect to the categories involved, which satisfy conditions stipulated in (1.1.4.b), namely those conditions given in (2.1.5.b).

(2.1.5) a. gap @ Sp = gap @ Mr = gap @ Hd = <n[ ]:X>

b. gap @ Sp = L1, gap @ Mr = gap @ Hd = L1$\oplus$L2

2.2 Constraints on Complementization

Thus we have to show how examples (2.1.1.a) and (2.1.1.c) are to be disallowed. For brevity of exposition, however, we will consider (2.2.1.a) and (2.2.1.b) instead.

(2.2.1) a. *[I wonder] which boy John met a girl who loved

b. *[John asked me] which boy Mary wondered who loved

The following treatment regarding syntax and semantics of relative constructions are quite informal, inadequate, and inaccurate. However, since our concern here is to show how island constraints are to be effected within a phrase structure account of English, any attempt to rectify this defect is bound to grow into something grossly out of proportion. Thus, let's simply assume that relative clauses are sanctioned through the following phrase structure rule. Here, all semantic representations are nothing more than notational junks, but if you would try to take the expressions of the form "gq(...)" as generalized quantifiers, you will see what I have in mind here.

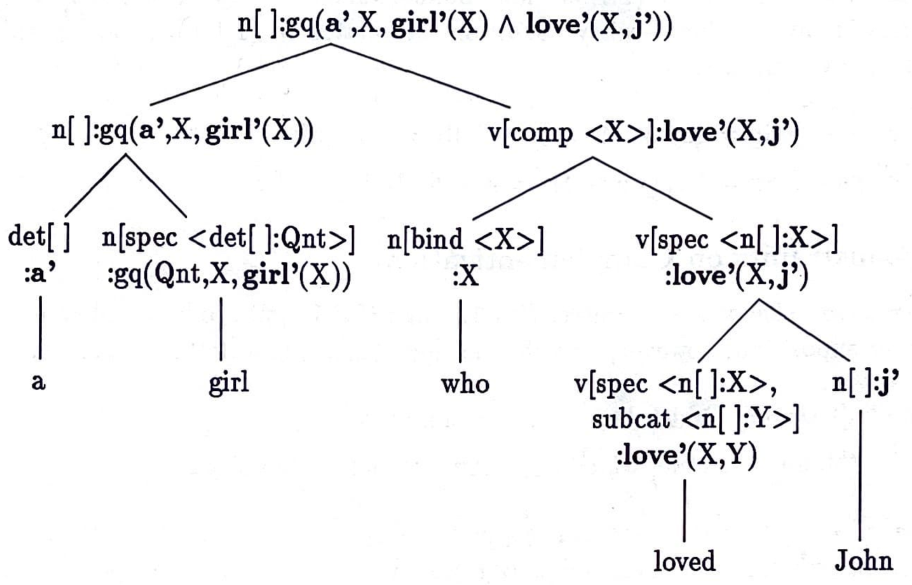
(2.2.2) adjunction 1

n[ ]:gq(Qnt,X,P1$\wedge$P2) $\rightarrow$ n[ ]:gq(Qnt,X,P1) v[comp <X>]:P2

This phrase structure rule is responsible for the following parse trees, among others.

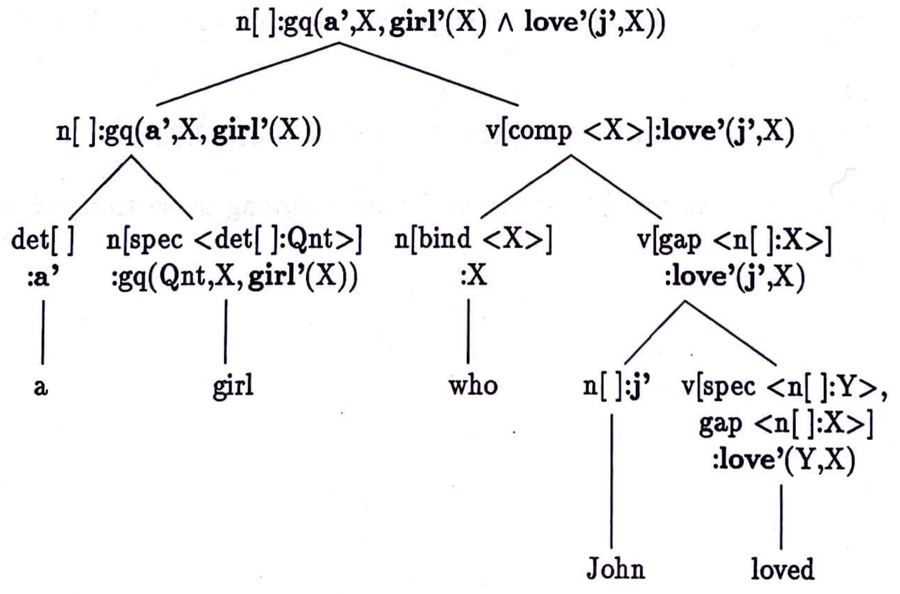
(2.2.3) a girl who loved John

gq(**a'**,X, **girl'**(X) ∧ **love'**(X,**j'**))


n[ ]:gq(a',X,girl'(X) ∧ love'(X,j'))
n[ ]:gq(a',X,girl'(X))
v[comp <X>]:love'(X,j')
det[ ] :a'
n[spec <det[ ]:Qnt>] :gq(Qnt,X,girl'(X))
n[bind <X>] :X
v[spec <n[ ]:X>] :love'(X,j')
a
girl
who
v[spec <n[ ]:X>, subcat <n[ ]:Y>] :love'(X,Y)
n[ ]:j'
loved
John

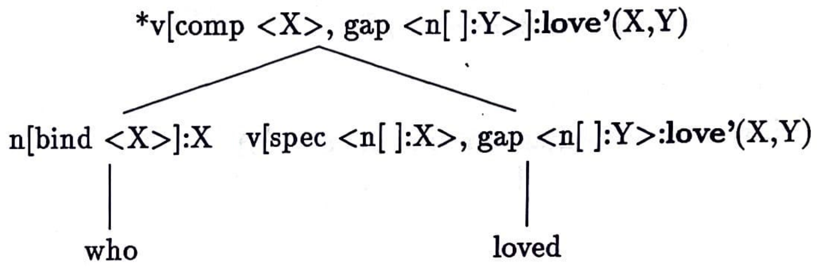
(2.2.4) a girl who John loved

gq(**a'**,X, **girl'**(X) ∧ **love'**(**j'**,X))


n[ ]:gq(a',X,girl'(X) ∧ love'(j',X))
n[ ]:gq(a',X,girl'(X))
v[comp <X>]:love'(j',X)
det[ ] :a'
n[spec <det[ ]:Qnt>] :gq(Qnt,X,girl'(X))
n[bind <X>] :X
v[gap <n[ ]:X>] :love'(j',X)
a
girl
who
n[ ]:j'
v[spec <n[ ]:Y>, gap <n[ ]:X>] :love'(Y,X)
John
loved

In order to see how (2.2.1.a) is disallowed, let's take a look at the following parse tree.

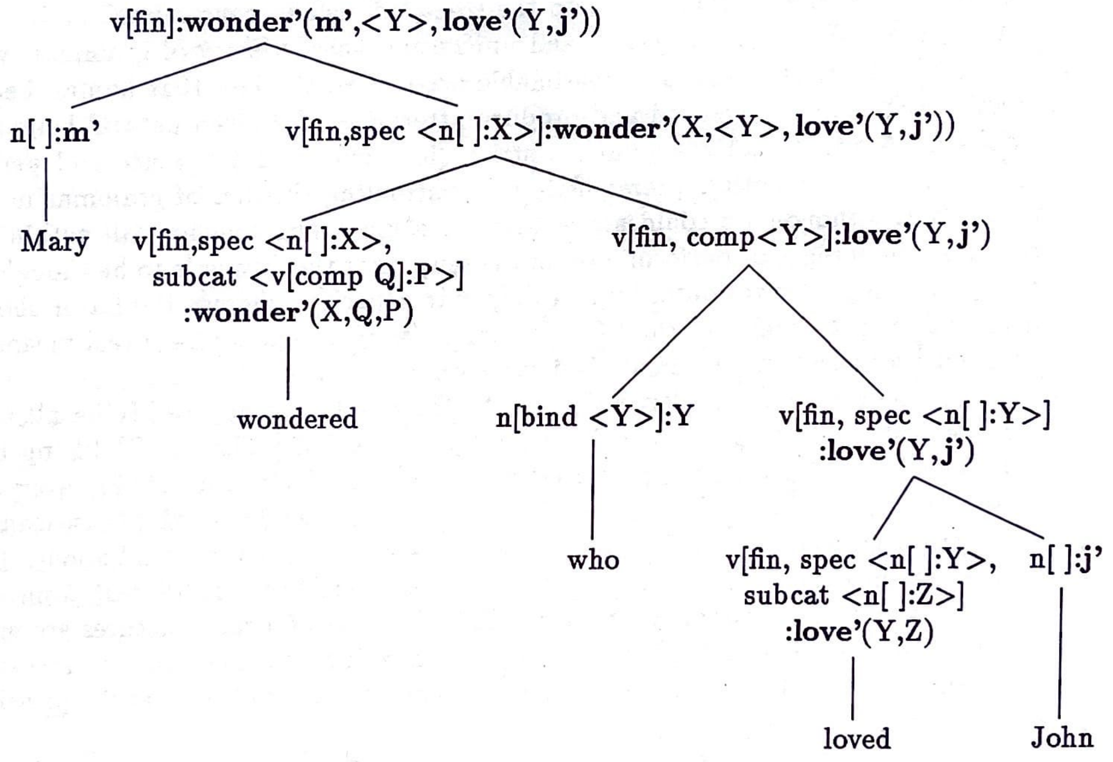
(2.2.5) *[I wonder] which boy John met a girl who loved


*v[comp <X>, gap <n[ ]:Y>]:**love'**(X,Y)
n[bind <X>]:X
v[spec <n[ ]:X>, gap <n[ ]:Y>:**love'**(X,Y)
who
loved

If this structure is to be admitted, complementization along with specification must be involved. However, since gap @ Mr is not nil, this is disallowed.

Likewise, embedded questions are assigned parse trees as shown in (2.2.6).

(2.2.6) Mary wondered who loved John
wonder'(m',<X>, love'(X,j'))

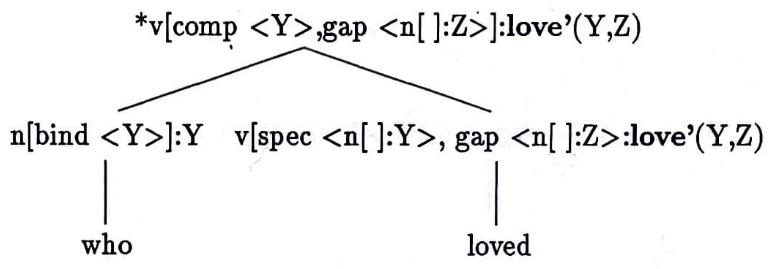

v[fin]:**wonder'(m',<Y>, love'(Y,j'))**
n[ ]:**m'**
v[fin,spec <n[ ]:X>]:**wonder'**(X,<Y>, **love'**(Y,j'))
Mary
v[fin,spec <n[ ]:X>, subcat <v[comp Q]:P>] :**wonder'**(X,Q,P)
v[fin, comp<Y>]:**love'**(Y,j')
wondered
n[bind <Y>]:Y
v[fin, spec <n[ ]:Y>] :**love'**(Y,j')
who
v[fin, spec <n[ ]:Y>, subcat <n[ ]:Z>] :**love'**(Y,Z)
n[ ]:j'
loved
John

Here we assume the following lexical entry.

(2.2.7) wondered $\models$ v[fin, spec<n[ ]:X>, subcat<v[comp Q]:P>]:**wonder'**(X,Q,P)

The ungrammaticality of (2.2.1.b) can be easily understood if you look at the following local phrase structure.

(2.2.8) *[John asked me] which boy Mary wondered who loved


*v[comp <Y>,gap <n[ ]:Z>]:love'(Y,Z)
n[bind <Y>]:Y
v[spec <n[ ]:Y>, gap <n[ ]:Z>:love'(Y,Z)
who
loved

Here again, we would like to have complementization along with specification, but this is not permitted since gap @ Mr is not nil.

3 Concluding Remarks

Phrase structure grammar descriptions of natural languages as shown above is not developed with giving explanation to the so-called "island constraints" as its central objective. It is a monostratal and unification based theory of grammar, with a particular interest in giving a reasonable account to the fact that human beings rapidly and easily understand and produce utterances of a given natural language. Unlike transformational grammars, a strict dichotomy of competence and performance is not a theoretical prerequisite in constructing theories of grammar in our framework, although we could argue whether a given phenomenon fall within the realm of competence or performance, in the sense that the former is to be thought of as a static characterization of human linguistic capacity, whereas the latter should involve dynamic aspects of this. It is to our advantage that we have real means to consider how these two are related to each other.

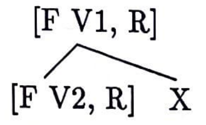
Our condition on complementization to the effect that gap @ Mr be nil is in a sense just a notational variant of FCR 20: ([SLASH]&[WH]) in GKPS pp.153-155. This feature cooccurrence restriction states, figuratively translated in everyday prose, that grammatical categories that are commanded by a wh-phrase cannot have gaps in them. This is in a sense what wh-island constraint is all about. Our constraint on complementization is, descriptively speaking, no different from this statement. However, we might try to predict what sort of local structures are more likely to be excluded in a given language by introducing a measure of processing complexity to our theory, while feature cooccurrence restrictions can in principle state any sort of stipulations.

Thus, our fragment of English utilized four category-valued features. Let us informally define the notion "operative in a given local phrase structure" with respect to these four category-valued features.

(3.1) operative in a local structure

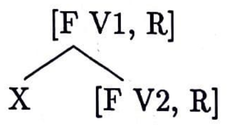
In the following local structures (a) and (b), F is operative.

a.



where length(V1) < length(V2)

b.



where length(V1) < length(V2)

Typically subcat is operative in complementation, spec in specification,gap in topicalization, bind in complementization. Note that complementization concurs syntactically with specification and topicalization. In these local structures, two category-valued features are operative. It is natural to assume that in performance processing of these structures is heavily loaded, thus dis llowing inheritance of non-null values of gap between the head and the mo her. Although this point has to be further attested through other than mere s eculations, this account is not unlikely to hold.

We notice that the following ordering relation hold among the four category-valued features introduced in the fragment above.

(3.2) subcat < spec < gap < bind

Here the expression "F1 < F2" states that in a local phrase structure in which F2 is operative, F1 @ Mr is to be nil. Note that since lexical rules do not involve phrase structures, this ordering is irrelevant in their application. If we incorporate this ordering into our theory of grammar, phrase structure rules in our grammar will be stated in much simpler forms. Finally, this relation seems quite natural if we take into consideration pre-theoretic significance of these features, that is subcat must be locally processed, or bind is semantically salient and so on. In this way, grammatical constraints might be in part or whole translated into processing complexity of a given configuration.

Acknowledgments

This article is based on a Japanese document I prepared for Unbounded Dependency Workshop Symposium on Island Constraints held at Tokyo Metropolitan University on May, 5, 1987. Although it was entitled "Island Constraints, from HPSG point of view" as the grammatical theory employed in that document resembled those unification-based theories exposed in Pollard (1984) and/or Pollard (1985) [HPSG] rather than those instantiation-based approaches of Gazdar and Pullum (1982) [GPSG-82] and/or Gazdar, Klein, Pullum and Sag (1985) [GKPS], the term "phrase structure grammar" is employed here because it does not really refer to differences among theories of GPSG. A slightly amended version of this Japanese document is available as Harada (1988).

I would like to express my gratitude to those responsible for the Unbounded Dependency Workshop, especially Professor Nakajima Heizo at Tokyo Metropolitan University, among others.

The grammatical description format employed in this article is sort of a by product of discussions with members of Japanese Phrase Structure Grammar Working Group at ICOT, whose chairman is Gunji Takao. I would like to express my thanks to all those involved in the project. Especially Hasida Kôiti and Sirai Hidetosi are at least as responsible for what's written here as I am. Some premature outbursts of ideas adopted here can be found in Harada (1986) and Harada (1987), if you prefer obscure explanations in English to accurate accounts in Japanese. For further details and developments of our approach, see Miyosi et. al. (1986) and Shirai et. al. (1987). Hasida and Shirai(1986) will show you what constraint unification is all about.

References

Gazdar, G. 1981. "Unbounded dependencies and coordinate structure." *Linguistic Inquiry*, **12**.

Gazdar, G. and G.K. Pullum. 1982. *Generalized Phrase Structure Grammar: a theoretical synopsis.* Bloomington: Indiana University Linguistics Club.

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

Harada, Y. 1986. "A Prolog implementation of SUHG." *Linguistic Research* No. 4, Tokyo University English Linguistics Association.

Harada, Y. 1987. "Toward grammatical descriptions of natural languages within extended unification environments," *Linguistic Research* No. 5, Tokyo University English Linguistics Association.

Harada, Y. 1988. "A phrase structure grammar account of island constraints (in Japanese)." *Humanitas*, No. 26, Tokyo: The Waseda University Law Association.

Hasida, K. and H. Sirai. 1986. "Constraint unification (in Japanese)." *Computer Software*, **3**, No. 4.

Miyosi, H., T. Gunji, H. Sirai, K. Hasida, and Y. Harada. 1986. "JPSG—a phrase structure grammar for Japanese (in Japanese)." *Computer Software*, Vol. 3, No. 4.

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

Pollard, C.J. 1985. "Lectures on HPSG." unpublished manuscript, Stanford University.

Sirai, H., T. Gunji, K. Hasida, and Y. Harada. 1987. "Grammatical descriptions with local constraints (in Japanese)." paper presented at Language Processing

and Communications Research Group, Institute for Electronics, Information, and Communication Engineers of Japan.

Received March 4, 1988