IN: THE MENTAL REPRESENTATION OF GRAMMATICAL RELATIONS
edited by Joan Bresnan
1982

Chapter 4

# Lexical-Functional Grammar: A Formal System for Grammatical Representation

Ronald M. Kaplan
Joan Bresnan

In learning their native language, children develop a remarkable set of capabilities. They acquire knowledge and skills that enable them to produce and comprehend an indefinite number of novel utterances and to make quite subtle judgments about certain of their properties. The major goal of psycholinguistic research is to devise an explanatory account of the mental operations that underlie these linguistic abilities.

In pursuing this goal, we have adopted what we call the *Competence Hypothesis* as a methodological principle. We assume that an explanatory model of human language performance will incorporate a theoretically justified representation of the native speaker's linguistic knowledge (a *grammar*) as a component separate both from the computational mechanisms that operate on it (a *processor*) and from other nongrammatical processing parameters that might influence the processor's behavior.[1] To a certain extent the various components that we postulate can be studied independently, guided where appropriate by the well-established methods and evaluation standards of linguistics. computer science, and experimental psychology. However, the requirement that the various components ultimately must fit together in a consistent and coherent model imposes even stronger constraints on their structure and operation.

This chapter presents a formalism for representing the native speaker's syntactic knowledge. In keeping with the Competence Hypothesis, this formalism, called *lexical-functional grammar* (LFG), has been designed to serve as a medium for expressing and explaining important generalizations about the syntax of human languages and thus to serve as a vehicle for independent linguistic research. Of equal significance, it is a restricted, mathematically tractable notation for which simple, psychologically plausible processing mechanisms can be defined. Lexical-functional grammar has evolved both from previous research within the transformational framework (e.g., Bresnan 1978) and from earlier computational and psycholinguistic investigations (Woods 1970, Kaplan 1972, 1973b, 1975b, Wanner and Maratsos 1978).

The fundamental problem for a theory of syntax is to characterize the mapping between semantic predicate-argument relationships and surface word and phrase configurations by which they are expressed. This mapping is sufficiently complex that it cannot be characterized in a simple, unadorned phrase structure formalism: a single set of predicate-argument relations can be realized in many different phrase structures (e.g., active and passive constructions), and a single phrase structure can express several different semantic relations, as in cases of ambiguity. In lexical-functional grammar, this correspondence is defined in two stages. Lexical entries specify a direct mapping between semantic arguments and configurations of surface grammatical functions. Syntactic rules then identify these surface functions with particular morphological and constituent structure configurations. Alternative realizations may result from alternative specifications at either stage of the correspondence. Moreover, grammatical specifications impose well-formedness conditions on both the functional and the constituent structures of sentences.

This chapter is concerned with the grammatical formalism itself; its linguistic, computational, and psychological motivation are dealt with in separate chapters and in other papers. In the next several sections we introduce the formal objects of our theory, discuss the relationships among them, and define the notation and operations for describing and manipulating them. Illustrations in these and later sections show possible LFG solutions to various problems of linguistic description. Section 4.5 considers the functional requirements that strings with valid constituent structures must satisfy. Section 4.6 summarizes arguments for the independence of the constituent, functional, and semantic levels of representation. In section 4.7 we introduce and discuss the formal apparatus for characterizing long-distance grammatical dependencies. We leave to the end the question of our system's generative power. We prove in section 4.8 that despite their linguistic expressiveness, lexical-functional grammars are *not* as powerful as unrestricted rewriting systems.

## 4.1 Constituent Structures and Functional Structures

A lexical-functional grammar assigns two levels of syntactic description to every sentence of a language. Phrase structure configurations are represented in a *constituent structure*. A constituent structure (or "c-structure") is a conventional phrase structure tree, a well-formed labeled bracketing that indicates the superficial arrangement of words and phrases in the sentence. This is the representation on which phonological interpretation operates to produce phonetic strings. Surface grammatical functions are represented explicitly at the other level of description, called *functional structure*. The functional structure ("f-structure") provides a precise characterization of such traditional syntactic notions as subject, "understood" subject, object, complement, and adjunct. The f-structure is the sole input to the semantic component, which may either translate the f-structure into the appropriate formulas in some logical language or provide an immediate model-theoretic interpretation for it.

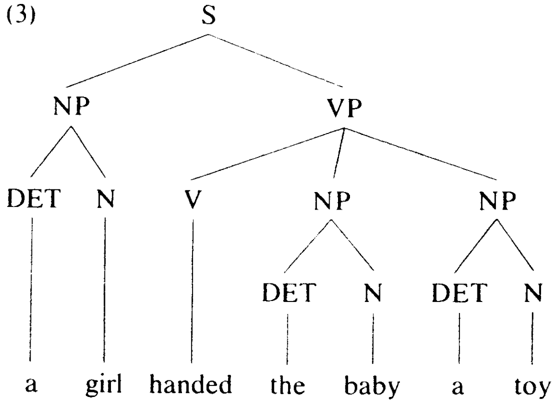
Constituent structures are formally quite different from functional structures. C-structures are defined in terms of syntactic categories, terminal strings, and their dominance and precedence relationships, whereas f-structures are composed of grammatical function names, semantic forms, and feature symbols. F-structures (and c-structures) are also distinct from semantic translations and interpretations, in which, for example, quantifier-scope ambiguities are resolved. By formally distinguishing these levels of representation, our theory attempts to separate those grammatical phenomena that are purely syntactic (involving only c-structures and f-structures) from those that are purely lexical (involving lexical entries before they are inserted into c-structures and f-structures) or semantic (for example, involving logical inference). Our framework thus facilitates an empirically motivated division of labor among the lexical, syntactic, semantic, and phonological components of a grammar.

A c-structure is determined by a grammar that characterizes all possible surface structures for a language. This grammar is expressed in a slightly modified context-free formalism or a formally equivalent specification such as a recursive transition network (Woods 1970, Kaplan 1972). For example, the ordinary rewriting procedure for context-free grammars would assign the c-structure (3) to the sentence (2), given the rules in (1):

(1)
a. S → NP VP
b. NP → DET N
c. VP → V NP NP

(2)
A girl handed the baby a toy.

(3)



We emphasize that c-structure nodes can be derived only by phrase structure rules such as (1a–c). There are no deletion or movement operations which could, for example, form the double-NP sequence from a phrase structure with a *to* prepositional phrase. Such mechanisms are unnecessary in LFG because we do not map between semantically and phonologically interpretable levels of phrase structure. Semantic interpretation is defined on functional structure, not on the phrase structure representation that is the domain of phonological interpretation.

The functional structure for a sentence encodes its meaningful grammatical relations and provides sufficient information for the semantic component to determine the appropriate predicate-argument formulas. The f-structure for (2) would indicate that the *girl* NP is the grammatical subject, *handed* conveys the semantic predicate, the *baby* NP is the grammatical object, and *toy* serves as the second grammatical object. The f-structure represents this information as a set of ordered pairs each of which consists of an *attribute* and a specification of that attribute's *value* for this sentence. An attribute is the name of a grammatical function or feature (SUBJ, PRED, OBJ, NUM, CASE, etc.). There are three primitive types of values:

(4)
a. Simple *symbols*
b. *Semantic forms* that govern the process of semantic interpretation
c. Subsidiary *f-structures*, sets of ordered pairs representing complexes of internal functions

A fourth type of value, *sets* of symbols, semantic forms, or f-structures, is also permitted. We will discuss this type when we consider the grammatical treatment of adjuncts.

Given possibility (4c), an f-structure is in effect a hierarchy of attribute/value pairs. We write an f-structure by arranging its pairs vertically inside square brackets with the attribute and value of a single pair placed on a horizontal line. The following is a plausible f-structure for sentence (2):

(5)

$$\left[\begin{array}{ll}
\text{SUBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`GIRL'}\end{array}\right] \\
\text{TENSE} & \text{PAST} \\
\text{PRED} & \text{`HAND}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ2})\ (\uparrow \text{OBJ})\rangle\text{'} \\
\text{OBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{THE}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`BABY'}\end{array}\right] \\
\text{OBJ2} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`TOY'}\end{array}\right]
\end{array}\right]$$

In this structure, the TENSE attribute has the simple symbol value PAST; pairs with this kind of value represent syntactic "features." Grammatical functions have subsidiary f-structure values, as illustrated by the subject function in this example:

(6)

$$\left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`GIRL'}\end{array}\right]$$

The attributes SPEC (specifier) and NUM mark embedded features with the symbol values A and SG, respectively.

The quoted values of the PRED attributes are semantic forms. Semantic forms usually arise in the lexicon[2] and are carried along by the syntactic component as unanalyzable atomic elements, just like simple symbols. When the f-structure is semantically interpreted, these forms are treated as patterns for composing the logical formulas encoding the meaning of the sentence. Thus, the semantic interpretation for this sentence is obtained from the value of its PRED attribute, the semantic form in (7):

(7)
'HAND⟨(↑ SUBJ) (↑ OBJ2) (↑ OBJ)⟩'

This is a predicate-argument expression containing the semantic predicate name HAND followed by an argument list specification enclosed in angle brackets. (The angle brackets correspond to the parentheses in the logical language that would ordinarily be used to denote the application of a predicate to its arguments. We use angle brackets in order to distinguish the semantic parentheses from the parentheses of our syntactic formalism.) The argument list specification defines a mapping between the logical or thematic arguments of the three-place predicate HAND (e.g., agent, theme, and goal) and the grammatical functions of the f-structure. The parenthetic expressions signify that the first argument position of that predicate is filled by the formula that results from interpreting the SUBJ function of the sentence, the formula from the OBJ2 is substituted in the second argument position, and so on. The formula for the embedded SUBJ f-structure is determined by *its* PRED value, the semantic form 'GIRL'. GIRL does not have an argument list because it does not apply to arguments specified by other grammatical functions. It is a predicate on individuals in the logical universe of discourse quantified by information derived from the SPEC feature.[3]

There are very strong compatibility requirements between a semantic form and the f-structure in which it appears. Loosely speaking, all the functions mentioned in the semantic form must be included in the f-structure, and all functions with subsidiary f-structure values must be mentioned in the semantic form. A given semantic form is in effect compatible with only one set of grammatical functions (although these may be associated with several different c-structures). Thus, the semantic form in (8) is not compatible with the grammatical functions in

(5) because it does not mention the OBJ2 function but does specify (↑ TO OBJ), the object of the preposition *to*.

(8)
'HAND⟨(↑ SUBJ) (↑ OBJ) (↑ TO OBJ)⟩'

This semantic form is compatible instead with the functions in the f-structure (9):

(9)

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`GIRL'} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \\ \text{PRED} & \text{`HAND}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ})\ (\uparrow \text{TO OBJ})\rangle\text{'} \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`TOY'} \end{bmatrix} \\ \text{TO} & \begin{bmatrix} \text{PCASE} & \text{TO} \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{THE} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`BABY'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

We show in section 4.4 how this f-structure is assigned to the NP–*to*–NP sentence (10):

(10)
A girl handed a toy to the baby.

This f-structure, with (8) as its PRED value, defines *girl, baby,* and *toy* as the agent, goal, and theme arguments of HAND, just as in (5). The native speaker's paraphrase intuitions concerning (2) and (10) are thus accurately expressed. This account of the English dative alternation is possible because our grammatical functions SUBJ, OBJ, TO OBJ, etc., denote *surface* grammatical relationships, not the underlying, logical relationships commonly represented in transformational deep structures.

The semantic forms (7) and (8) are found in alternative entries of the lexical item *handed*, reflecting the fact that the predicate HAND permits the alternative surface realizations (2) and (10), among others. Of course, many other verbs in the lexicon are similar to *handed* in having separate entries along the lines of (7) and (8). Our theory captures the systematic connection between NP–NP and NP–*to*–NP constructions by means of a lexical redundancy rule of the sort suggested by Bresnan (see Bresnan 1978 and chapter 1 of this volume). The semantic form (7) results from applying the "dativizing" lexical rule shown in (11) to the semantic form in (8).

(11)
(↑ OBJ) → (↑ OBJ2)
(↑ TO OBJ) → (↑ OBJ)

According to this rule, a word with a lexical entry containing the specifications (↑ OBJ) and (↑ TO OBJ) may have another entry in which (↑ OBJ2) appears in place of (↑ OBJ) and (↑ OBJ) appears in place of (↑ TO OBJ).

It is important to note that these relation-changing rules are not applied in the syntactic derivation of individual sentences. They merely express patterns of redundancy that obtain among large but finite classes of lexical entries and presumably simplify the child's language acquisition task (see chapter 10 for discussion). Indeed, just as our formalism admits no rules for transforming c-structures, it embodies a similar prohibition against syntactic manipulations of function assignments and function/argument mappings:

(12)
*Direct Syntactic Encoding*
No rule of syntax may replace one function name by another.

This principle is an immediate consequence of the Uniqueness Condition, which is stated in the next section. The principle of direct syntactic encoding sharpens the distinction between two classes of rules: rules that change relations are lexical and range over finite sets, while syntactic rules that project onto an infinite set of sentences preserve grammatical relations.[4] Our restrictions on the expressive power of syntactic rules guarantee that a sentence's grammatical functions are "visible" directly in the surface structure and thus afford certain computational and psychological advantages.

## 4.2 Functional Descriptions

A string's constituent structure is generated by a context-free c-structure grammar. That grammar is augmented so that it also produces

a finite collection of statements specifying various properties of the string's f-structure. The set of such statements, called the *functional description* ("f-description") of the string, serves as an intermediary between the c-structure and the f-structure.

The statements of an f-description can be used in two ways. They can be applied to a particular f-structure to decide whether or not it has all the properties required by the grammar. If so, the candidate f-structure may be taken as the f-structure that the grammar assigns to the string. The f-description may also be used in a constructive mode: the statements support a set of inferences by which an f-structure satisfying the grammar's requirements may be synthesized. The f-description is thus analogous to a set of simultaneous equations in elementary algebra that express properties of certain unknown numbers. Such equations may be used to validate a proposed solution, or they may be solved by means of arithmetic inference rules (canceling, substitution of equals for equals, etc.) to discover the particular numbers for which the equations are true. In line with this analogy, this section presents an algebraic formalism for representing an f-description.

The statements in an f-description and the inferences that may be drawn from them depend crucially on the following axiom:

(13)
*Uniqueness*
In a given f-structure, a particular attribute may have at most one value.

This condition makes it possible to describe an f-structure by specifying *the* (unique) values of the grammatical functions of which it is composed. Thus, if we let the variables $f_1$ and $f_2$ stand for unknown f-structures, the following statements have a clear interpretation:

(14)
a. the SUBJ of $f_1 = f_2$
b. the SPEC of $f_2$ = A
c. the NUM of $f_2$ = SG
d. the PRED of $f_2$ = 'GIRL'

In fact, these statements are true if $f_1$ and $f_2$ are the f-structures (5) and (6), and the statements in (14) may thus be considered a part of the f-description of sentence (2).

We have defined a functional structure as a set of ordered pairs satisfying the Uniqueness Condition (13). We now observe that this is precisely the standard definition of a *mathematical* function. There is a systematic ambiguity in our use of the word *function:* an f-structure is a mathematical function that represents the *grammatical* functions of a sentence. This coincidence provides a more conventional terminology for formulating the statements of an f-description. For example, statement (14c) can be paraphrased as (15a), and this can be stated more formally using the familiar parenthesis notation to indicate the application of a function to an argument, as in (15b):

(15)
a. The function $f_2$ is such that applying it to the argument NUM yields the value SG.
b. $f_2(\text{NUM}) = \text{SG}$

Thus, the statements of an f-description are simply equations that describe the values obtained by various function applications. Unlike the typical functions of elementary algebra, an f-structure is a function with a finite domain and range and thus can be defined by a finite table of arguments and values, as represented in our square bracket notation. Also, we do not draw a clear distinction between functions and their values. Algebraic equations commonly involve a known function that takes on a given value when applied to some unknown argument; the problem is to determine that argument. In (15b), however, the argument and the corresponding value are *both* known, and the problem is to find the function![5] Moreover, applying an f-structure to an argument may produce a function that may be applied in turn to another argument. If (16a) is true, then the stipulations in (15b) and (16b) are equivalent.

(16)
a. $$f_1(\text{SUBJ}) = \begin{bmatrix} \text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`GIRL'} \end{bmatrix} = f_2$$
b. $f_1(\text{SUBJ})(\text{NUM}) = \text{SG}$

The form of function composition illustrated in equation (16b) occurs quite often in f-descriptions. We have found that a slight adaptation of the traditional notation improves the readability of such specifications. Thus, we denote a function application by writing the function name *inside* the parentheses next to the argument instead of putting it in front. In our modified notation, the stipulation (15b) is written as (17a) and the composition (16b) appears as (17b).

(17)
a. $(f_2\ \text{NUM}) = \text{SG}$
b. $((f_1\ \text{SUBJ})\ \text{NUM}) = \text{SG}$

We make one further simplification: since all f-structures are functions of one argument, parenthetic expressions with more than two elements (a function and its argument) do not normally occur. Thus, we introduce no ambiguity by defining our parenthesis notation to be left-associative, by means of the identity (18):

(18)
$((f\ \alpha)\ \beta) \equiv (f\ \alpha\ \beta)$

This allows any leftmost pair of parentheses to be removed (or inserted) when convenient, so that (17b) may be simplified to (19):

(19)
$(f_1\ \text{SUBJ NUM}) = \text{SG}$

With this notation, there is a simple way of determining the value of a given function-application expression: we locate the f-structure denoted by the leftmost element in the expression and match the remaining elements from left to right against successive attributes in the f-structure hierarchy. Also, the English genitive construction provides a natural gloss for these expressions: (19) may be read as "$f_1$'s SUBJ's NUM is SG."

### 4.3 From C-Structures to F-Descriptions

Having said what an f-description is, we now consider how the f-description for a string is produced from a grammar and lexicon. This is followed by a discussion of the inferences that lead from an f-description to the f-structure that it describes.

The statements in an f-description come from functional specifications that are associated with particular elements on the righthand sides of c-structure rules and with particular categories in lexical entries. These specifications consist of templates from which the f-description statements are derived. A template, or statement *schema*, has the form of the statement to be derived from it except that in place of f-structure variables it contains special *metavariables*. If a rule is applied to generate a c-structure node or a lexical item is inserted under a preterminal category, the associated schemata are *instantiated* by replacing the metavariables with actual variables ($f_1, f_2, \ldots$). Which actual variables are used depends on which metavariables are in the schemata and what the node's relationship is to other nodes in the tree. The metavariables and grammatically significant tree relations are of just two types:

(20)
*Immediate domination*, with metavariables $\uparrow$ and $\downarrow$
*Bounded domination*, with metavariables $\Uparrow$ and $\Downarrow$

Statements based on nonimmediate but bounded tree relations are needed to characterize the "long-distance" dependencies found in relative clauses, questions, and other constructions. We postpone our discussion of bounded domination to section 4.7, since it is more complex than immediate domination.

Schemata involving immediate domination metavariables and relations yield f-description statements defining the local predicate-argument configurations of simple sentence patterns such as the dative. To illustrate, the c-structure rules (21a,b,c) are versions of (1a,b,c) with schemata written beneath the rule elements that they are associated with.

(21)

a. $\mathrm{S} \rightarrow \underset{(\uparrow \text{SUBJ})=\downarrow}{\mathrm{NP}} \quad \underset{\uparrow=\downarrow}{\mathrm{VP}}$

b. $\mathrm{NP} \rightarrow \mathrm{DET}\ \mathrm{N}$

c. $\mathrm{VP} \rightarrow \mathrm{V} \quad \underset{(\uparrow \text{OBJ})=\downarrow}{\mathrm{NP}} \quad \underset{(\uparrow \text{OBJ2})=\downarrow}{\mathrm{NP}}$

According to the instantiation procedure described below, the SUBJ and OBJ schemata in this grammar indicate that the subject and object f-structures come from NPs immediately dominated by S and VP. While superficially similar to the standard transformational definitions of *subject* and *object* (Chomsky 1965), our specifications apply only to surface constituents and establish only a loose coupling between functions and phrase structure configurations. Given the OBJ2 schema, for example, an NP directly dominated by VP can also function as a second object. These schemata correspond more closely to the SETR operation of the augmented transition network (ATN) notation (Woods 1970): ($\uparrow$ SUBJ) = $\downarrow$ has roughly the same effect as the ATN action (SETR SUBJ *). The direct equality on the VP category in (21a) has no ATN (or transformational) equivalent, however. It is an *identification* schema indicating that a single f-structure is based on more than one constituent, and thus that the f-structure is somewhat "flatter" than the c-structure.

The syntactic features and semantic content of lexical items are determined by schemata in lexical entries. The entries for the vocabulary of sentence (2) are listed in (22). (This illustration ignores the morphological composition of lexical items, which makes a systematic contribution to the set of inflectional features represented in the schemata.)

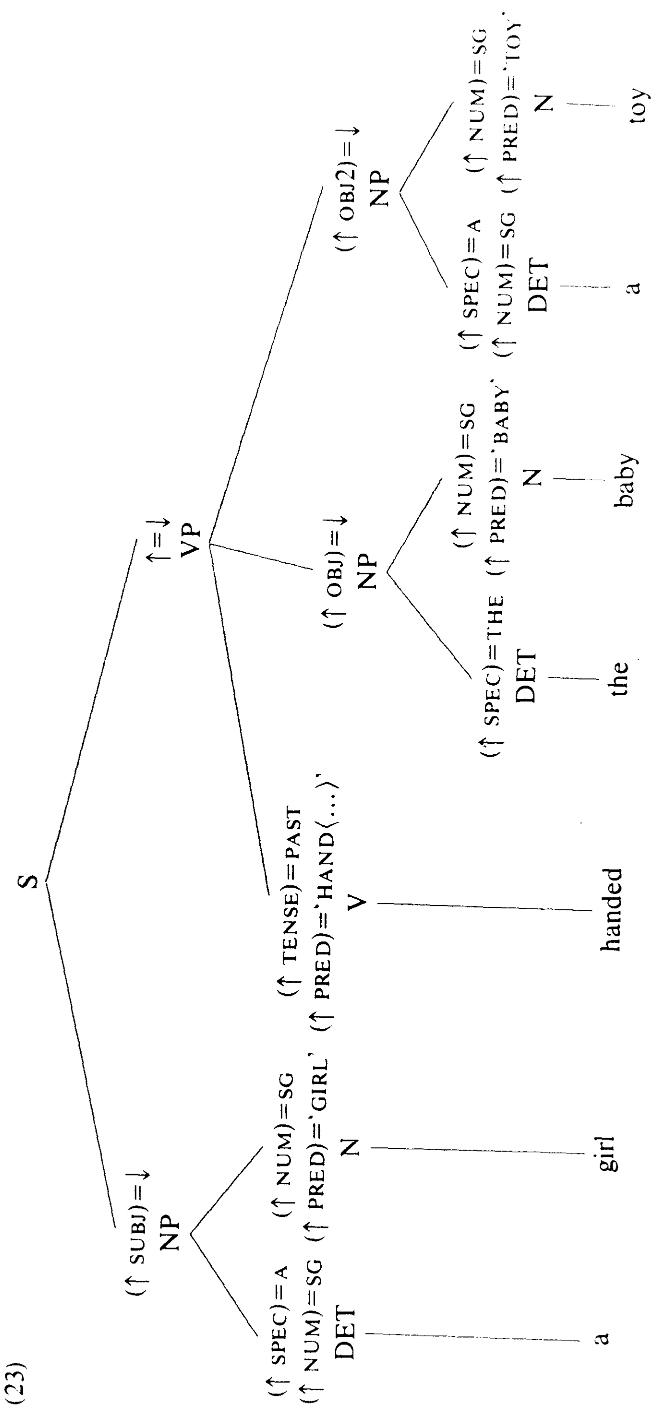
(22)

| | | |
|---|---|---|
| a: | DET, | ($\uparrow$ SPEC) = A |
| | | ($\uparrow$ NUM) = SG |
| girl: | N, | ($\uparrow$ NUM) = SG |
| | | ($\uparrow$ PRED) = 'GIRL' |
| handed: | V, | ($\uparrow$ TENSE) = PAST |
| | | ($\uparrow$ PRED) = 'HAND⟨($\uparrow$ SUBJ) ($\uparrow$ OBJ2) ($\uparrow$ OBJ)⟩' |
| the: | DET, | ($\uparrow$ SPEC) = THE |
| baby: | N, | ($\uparrow$ NUM) = SG |
| | | ($\uparrow$ PRED) = 'BABY' |
| toy: | N, | ($\uparrow$ NUM) = SG |
| | | ($\uparrow$ PRED) = 'TOY' |

A lexical entry in LFG includes a categorial specification indicating the preterminal category under which the lexical item may be inserted, and a set of schemata to be instantiated. As shown in (22), schemata originating in the lexicon are not formally distinct from those coming from c-structure rules, and they are treated uniformly by the instantiation procedure.

Instantiation is carried out in three phases. The schemata are first attached to appropriate nodes in the c-structure tree, actual variables are then introduced at certain nodes, and finally those actual variables are substituted for metavariables to form valid f-description statements. In the first phase, schemata associated with a c-structure rule element are attached to the nodes generated by that element. Lexical schemata are considered to be associated with a lexical entry's categorial specification and are thus attached to the nodes of that category that dominate the lexical item.[6] Attaching the grammatical and lexical schemata in (21) and (22) to the c-structure for sentence (2) produces the structure shown in (23). In this example we have written the schemata above the nodes they are attached to.

(23)



In the second phase of the instantiation procedure, a new actual variable is introduced for the root node of the tree and for each node where a schema contains the ↓ metavariable. Intuitively, the existence of ↓ at a node means that one component of the sentence's f-structure corresponds to that subconstituent. The new variable, called the *↓-variable* of the node, is a device for describing the internal properties of that f-structure (called the node's *↓ f-structure*) and its role in larger structures. In (24) we have associated ↓-variables with the nodes as required by the schemata in (23).

With the schemata and variables laid out on the tree in this way, the substitution phase of instantiation is quite simple. Fully instantiated statements are formed by substituting a node's ↓-variable first for all the ↓'s at that node and then for all the ↑'s attached to the nodes it immediately dominates. Thus, arrows pointing toward each other across one line in the tree are instantiated with the same variable.[7] The ↑ is called the "mother" metavariable, since it is replaced by the ↓-variable of its mother node. From the point of view of the S-dominated NP node, the schema (↑ SUBJ) = ↓ may be read as 'My mother's f-structure's SUBJ is my f-structure'.[8] In this case, the mother's variable is the root node's ↓-variable and so represents the f-structure of the sentence as a whole.

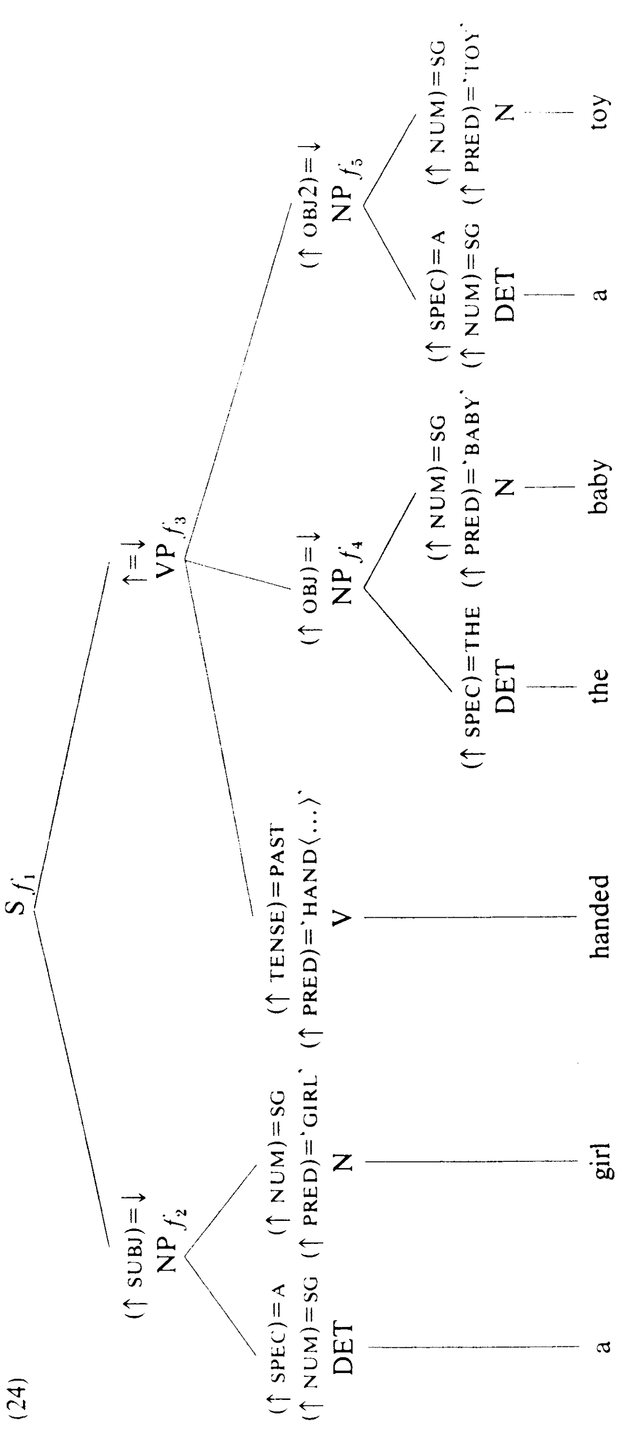
When we perform the substitutions for the schemata and variables in (24), the schemata attached to the S-dominated NP and VP nodes yield the equations in (25), and the daughters of the VP cause the equations in (26) to be included in the sentence's f-description:

(25)
a. $(f_1 \text{ SUBJ}) = f_2$
b. $f_1 = f_3$

(26)
a. $(f_3 \text{ OBJ}) = f_4$
b. $(f_3 \text{ OBJ2}) = f_5$

The equations in (25)–(26) taken together constitute the syntactically determined statements of the sentence's functional description. The other equations in the f-description are derived from the schemata on the preterminal nodes.

(24)



(27)

a. ($f_2$ SPEC) = A — from *a*
b. ($f_2$ NUM) = SG

c. ($f_2$ NUM) = SG — from *girl*
d. ($f_2$ PRED) = 'GIRL'

e. ($f_3$ TENSE) = PAST — from *handed*
f. ($f_3$ PRED) = 'HAND⟨(↑SUBJ) (↑OBJ2) (↑OBJ)⟩'

g. ($f_4$ SPEC) = THE — from *the*

h. ($f_4$ NUM) = SG — from *baby*
i. ($f_4$ PRED) = 'BABY'

j. ($f_5$ SPEC) = A — from *a*
k. ($f_5$ NUM) = SG

l. ($f_5$ NUM) = SG — from *toy*
m. ($f_5$ PRED) = 'TOY'

(For simplicity in this chapter, we do not instantiate the ↑ metavariable when it appears within semantic forms. This is permissible because the internal structure of semantic forms is not accessible to syntactic rules. However, the semantic translation or interpretation procedure may depend on a full instantiation.) Adding (27a–m) to the equations in (25)–(26) gives the complete f-description for sentence (2).

### 4.4 From F-Descriptions to F-Structures

Once an f-description has been produced for a given string, algebraic manipulations can be performed on its statements to make manifest certain implicit relationships that hold among the properties of that string's f-structure. These manipulations are justified by the left-associativity of the function-application notation (18) and by the substitution axiom for equality. To take an example, the value of the number feature of sentence (2)'s f-structure (that is, the value of ($f_1$ OBJ NUM)) can be inferred in the following steps:

(28)

| | | |
|---|---|---|
| ($f_1$ OBJ NUM) | = ($f_3$ OBJ NUM) | Substitution using (25b) |
| | = (($f_3$ OBJ) NUM) | Left-associativity |
| | = ($f_4$ NUM) | Substitution using (26a) |
| | = SG | Substitution using (27h) |

An f-description also supports a more important set of inferences: the equations can be "solved" by means of a construction algorithm that actually builds the f-structure they describe.

An f-structure solution may not exist for every f-description, however. If the f-description stipulates two distinct values for a particular attribute, or if it implies that an attribute name is an f-structure or semantic form instead of a symbol, then its statements are inconsistent with the basic axioms of our theory. In this case we classify the string as syntactically ill-formed, even though it has a valid c-structure. The functional well-formedness conditions of our theory thus account for many types of ungrammaticality. It is therefore essential that there be an algorithm for deciding whether or not an f-description is consistent, and for producing a consistent f-description's f-structure solution. Otherwise, our grammars would generate all but not *only* the sentences of a language.

Fortunately, f-descriptions are well-understood mathematical objects. The problem of determining whether or not a given f-description is satisfiable is equivalent to the decision problem of the quantifier-free theory of equality. Ackermann 1954 proved that this problem is solvable, and several efficient solution algorithms have been discovered (for example, the congruence closure algorithm of Oppen and Nelson 1977). In this section we outline a decision and construction algorithm whose operations are specially adapted to the linguistic representations of our theory.

We begin by giving a more precise interpretation for the formal expressions that appear in f-description statements. We imagine that there is a collection of entities (symbols, semantic forms, and f-structures) that an f-description characterizes, and that each of these entities has a variety of names, or *designators*, by which the f-description may refer to it. The character strings that we have used to represent symbols and semantic forms, the algebraic variables we introduce, and the function-application expressions are all designators. The entity denoted by a designator is called its *value*. The value of a symbol or semantic form character string is obviously the identified symbol or semantic form. The value of a variable designator is of course not obvious from the variable's spelling; it is defined by an assignment list of variable–entity pairs. A basic function-application expression is a parenthesized pair of designators, and its value is the entity, if any, obtained by applying the f-structure value of the left designator to the symbol value of the right designator.[9] This rule applies recursively if either expression

is itself a function-application: to obtain the value of $((f_1$ OBJ) NUM), we must first obtain the value of $(f_1$ OBJ) by applying the value of $f_1$ to the symbol OBJ.

Note that several different designators may refer to the same entity. The deduction in (28), for example, indicates that the designators $(f_1$ OBJ NUM) and $(f_4$ NUM) both have the same value, the symbol SG. Indeed, we interpret the equality relation between two designators as an explicit stipulation that those designators name the same entity. In processing an f-description, our algorithm attempts to find a way of associating with designators values that are consistent with the synonymy relation implied by the equality statements and with the procedure just outlined for obtaining the values of different types of designators.

The algorithm works by successive approximation.[10] It goes through a sequence of steps, one for each equation in the f-description. At the beginning of each step, it has a collection of symbols, semantic forms, and f-structures that satisfy all the equations considered at preceding steps, together with an assignment of tentative values for the variables occurring in those equations. The algorithm revises the collection of entities and value assignments to satisfy in addition the requirements of one more equation from the f-description. The entities after the last equation is processed thus satisfy the f-description as a whole and provide a final value for the ↓-variable of the c-structure tree's root node. This is the f-structure that the grammar assigns to the string.

The processing of a single equation is carried out by means of two operators. One operator, called *Locate*, obtains the value for a given designator. The entities in the collection might be augmented by the Locate operator to ensure that a value exists for that designator. When the values for the lefthand and righthand designators have been located, the second operator, *Merge*, checks to see whether those values are the same and hence already satisfy the equality relation. If not, it constructs a new entity by combining the properties of the distinct values, provided those properties are compatible. The collection is revised so that this entity becomes the common value of the two designators and also of all previously encountered synonyms of these designators. Stated in more formal terms, if $d_1$ and $d_2$ are the designators in an equation $d_1 = d_2$, and if brackets represent the application of an operator to its arguments, then that equation is processed by performing Merge[Locate[$d_1$], Locate[$d_2$]].

A technical definition of these operators is given in the appendix. In this section we present an intuitive description of the solution process, using as an example the f-description in (25)–(27). The final result does not depend on the order in which equations are considered, so we will simply take them as they appear above. We start with an empty collection of entities and consider equation (25a): $(f_1 \text{ SUBJ}) = f_2$. To locate the value of $(f_1 \text{ SUBJ})$, we must first obtain the value of $f_1$. There is as yet no assignment for that variable, so the Locate operator creates a value out of whole cloth: it adds a special "place-holder" entity to our collection and assigns it as the value of $f_1$. A representation for the new entity and variable assignment is shown in (29):

(29)

$$f_1 \ \rule{1cm}{0.4pt}$$

A place-holder is represented by a blank line, indicating that it is an entity none of whose properties are known. The variable prefix signifies that whatever that entity is, it has been assigned as the tentative value of $f_1$. A place-holder is just a bookkeeping device for recording the relations between entities before we have discovered anything else about them.

With the value of $f_1$ in hand, we return to the larger designator $(f_1 \text{ SUBJ})$. This provides more specific information about the entity that the place-holder stands for: the value of $f_1$ must be an f-structure that has SUBJ as one of its attributes. We revise our collection again to take account of this new information:

(30)

$$f_1\left[\begin{array}{ll}\text{SUBJ} & \rule{1cm}{0.4pt}\end{array}\right]$$

Knowing nothing about the value of SUBJ in the $f_1$ f-structure, we have represented it by another place-holder. This place-holder is the entity located for the designator $(f_1 \text{ SUBJ})$. We now turn to $f_2$, the second designator in the equation. This is a variable with no previous assignment, so our location procedure simply assigns it to another newly created place-holder:

(31)

$$f_2 \ \rule{1cm}{0.4pt}$$

This completes the location phase of the algorithm's first step: the equation's designators now denote the place-holders in (30) and (31).

The Merge operator changes the collection once more, so that the two designators denote the same entity. The two place-holders are distinct, but neither has any properties. Thus, a common value, also a place-holder with no properties, can be constructed. This place-holder appears as the value of SUBJ in the $f_1$ f-structure, but it is also assigned as the value of $f_2$, as shown in (32):

(32)

$$f_1\left[\begin{array}{ll}\text{SUBJ} & f_2 \ \rule{1cm}{0.4pt}\end{array}\right]$$

The structure (32) is now the only member of our entity collection. Notice that with this assignment of variables, the designators $(f_1 \text{ SUBJ})$ and $f_2$ have the same value, so the equation $(f_1 \text{ SUBJ}) = f_2$ is satisfied.

We move on to equation (25b), the identification $f_1 = f_3$. This means that the variables $f_1$ and $f_3$ are two different designators for a single entity. That entity will have all the properties ascribed via the designator $f_1$ and also all the properties ascribed to the synonymous $f_3$. The f-structure (32) is located as the value of $f_1$, and a new place-holder is assigned to $f_3$. Since the place-holder has no properties, the result of combining it with the f-structure is simply that f-structure again, with its variable prefixes modified to reflect the new equality. Thus, the result of the merge for the second equation is (33):

(33)

$$f_1\ f_3\left[\begin{array}{ll}\text{SUBJ} & f_2 \ \rule{1cm}{0.4pt}\end{array}\right]$$

The variable assignments in (33) now satisfy the first two equations of the f-description.

The equation at the next step is (26a): $(f_3 \text{ OBJ}) = f_4$. $f_3$ already has an f-structure value in (33), but it does not include OBJ as one of its attributes. This is remedied by adding an appropriate place-holder:

(34)

$$f_1\ f_3\left[\begin{array}{ll}\text{SUBJ} & f_2 \ \rule{1cm}{0.4pt} \\ \text{OBJ} & \quad \rule{1cm}{0.4pt}\end{array}\right]$$

This place-holder is merged with one created for the variable $f_4$, yielding (35):

(35)

$$
{}_{f_1\,f_3}\left[\begin{array}{ll}
\text{SUBJ} & f_2\ \rule{2em}{0.4pt} \\
\text{OBJ} & f_4\ \rule{2em}{0.4pt}
\end{array}\right]
$$

Equation (26b) is handled in a similar fashion and results in (36):

(36)

$$
{}_{f_1\,f_3}\left[\begin{array}{ll}
\text{SUBJ} & f_2\ \rule{2em}{0.4pt} \\
\text{OBJ} & f_4\ \rule{2em}{0.4pt} \\
\text{OBJ2} & f_5\ \rule{2em}{0.4pt}
\end{array}\right]
$$

After we have processed these equations, our collection of entities and variable assignments satisfies all the syntactically determined equations of the f-description.

The lexically derived equations are now taken into account. These have the effect of adding new features to the outer f-structure and filling in the internal properties of the place-holders. Locating the value of the lefthand designator in equation (27a), ($f_2$ SPEC) = A, converts the SUBJ place-holder to an f-structure with a SPEC feature whose value is a new place-holder:

(37)

$$
{}_{f_1\,f_3}\left[\begin{array}{ll}
\text{SUBJ} & {}_{f_2}\left[\begin{array}{ll}\text{SPEC} & \rule{2em}{0.4pt}\end{array}\right] \\
\text{OBJ} & f_4\ \rule{2em}{0.4pt} \\
\text{OBJ2} & f_5\ \rule{2em}{0.4pt}
\end{array}\right]
$$

The value of the righthand designator is just the symbol A. Merging this with the new SPEC place-holder yields (38):

(38)

$$
{}_{f_1\,f_3}\left[\begin{array}{ll}
\text{SUBJ} & {}_{f_2}\left[\begin{array}{ll}\text{SPEC} & \text{A}\end{array}\right] \\
\text{OBJ} & f_4\ \rule{2em}{0.4pt} \\
\text{OBJ2} & f_5\ \rule{2em}{0.4pt}
\end{array}\right]
$$

Note that this modification does not falsify any equations processed in previous steps.

Equation (27b) has the same form as (27a), and its effect is simply to add a SG-valued NUM feature to the SUBJ f-structure, alongside the SPEC:

(39)

$$
{}_{f_1\,f_3}\left[\begin{array}{ll}
\text{SUBJ} & {}_{f_2}\left[\begin{array}{ll}\text{SPEC} & \text{A} \\ \text{NUM} & \text{SG}\end{array}\right] \\
\text{OBJ} & f_4\ \rule{2em}{0.4pt} \\
\text{OBJ2} & f_5\ \rule{2em}{0.4pt}
\end{array}\right]
$$

Though derived from different lexical items, equation (27c) is an exact duplicate of (27b). Processing this equation therefore has no visible effects.

The remaining equations are quite straightforward. Equation (27d) causes the PRED function to be added to the SUBJ f-structure, (27e)–(27f) yield the TENSE and PRED functions in the $f_1$–$f_3$ structure, and (27g)–(27m) complete the OBJ and OBJ2 place-holders. Equation (27l) is similar to (27c) in that it duplicates another equation in the f-description and hence does not have an independent effect on the final result. After considering all the equations in (27), we arrive at the final f-structure (40):

(40)

$$
{}_{f_1\,f_3}\left[\begin{array}{ll}
\text{SUBJ} & {}_{f_2}\left[\begin{array}{ll}\text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`GIRL'}\end{array}\right] \\
\text{OBJ} & {}_{f_4}\left[\begin{array}{ll}\text{SPEC} & \text{THE} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`BABY'}\end{array}\right] \\
\text{OBJ2} & {}_{f_5}\left[\begin{array}{ll}\text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`TOY'}\end{array}\right] \\
\text{TENSE} & \text{PAST} \\
\text{PRED} & \text{`HAND}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ2})\ (\uparrow \text{OBJ})\rangle\text{'}
\end{array}\right]
$$

Since $f_1$ is the ↓-variable of the root node of the tree (24), the outer f-structure is what our simple grammar assigns to the string. This is just the structure in (5), if the variable prefixes and the order of pairs are ignored.

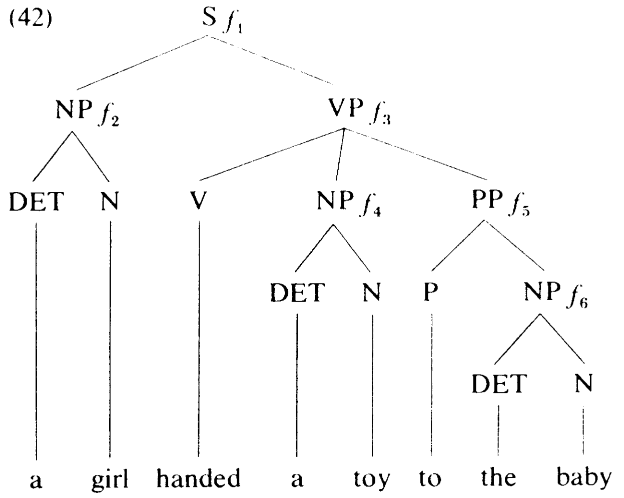
This example is special in that the argument positions of all the function-application designators are filled with symbol designators. Certain grammatical situations give rise to less restricted designators, where the argument position is filled with another function-application. This is possible because symbols have a dual status in our formalism: they can serve in an f-structure both as attributes and as values. These more general designators permit the grammatical relation assigned to the ↓ f-structure at a given node to be determined by internal features of that f-structure rather than by the position of that node in the c-structure. The arguments to a large number of English verbs, for instance, may appear as the objects of particular prepositions instead of as SUBJ, OBJ, or OBJ2 noun phrases. In our theory, the lexical entry for a "case-marking" preposition indicates that its object noun phrase may be treated as what has traditionally been called a verb's *oblique object*. The semantic form for the verb then specifies how to map that oblique object into the appropriate argument of the predicate.

The *to* alternative for the double-NP realization of *handed* provides a simple illustration. The contrasting sentence to our previous example (2) is (10), repeated here for convenience:

(41)
A girl handed a toy to the baby.

The c-structure for this sentence with a set of ↓-variables for the functionally relevant nodes is shown in (42). It includes a prepositional phrase following the object NP, as permitted by the new c-structure rules (43a,b). (We use the standard context-free abbreviation for optionality, parentheses that enclose categories and schemata. Thus, (43a) also derives intransitive and transitive verb phrases. Optionality parentheses should not be confused with the function-application parentheses within schemata. We also use braces in rules to indicate alternative c-structure expansions.)

(42)



(43)

a. $$\mathrm{VP} \rightarrow \mathrm{V} \begin{pmatrix} \mathrm{NP} \\ (\uparrow \mathrm{OBJ})=\downarrow \end{pmatrix} \begin{pmatrix} \mathrm{NP} \\ (\uparrow \mathrm{OBJ2})=\downarrow \end{pmatrix} \begin{matrix} \mathrm{PP}^* \\ (\uparrow (\downarrow \mathrm{PCASE}))=\downarrow \end{matrix}$$

b. $$\mathrm{PP} \rightarrow \mathrm{P} \quad \begin{matrix} \mathrm{NP} \\ (\uparrow \mathrm{OBJ})=\downarrow \end{matrix}$$

The PP element in (43a) exhibits two new rule features. The asterisk on the PP category symbol is the Kleene-star operator; it indicates that that rule element may be repeated any number of times, including none.[11] The schema on the PP specifies that the value of the PCASE attribute in the PP's f-structure determines the functional role assigned to that structure. Because the lexical schemata from *to* are attached to the P node, that feature percolates up to the f-structure at the PP node. Suppose that *to* has the case-marking lexical entry shown in (44a)[12] and that *handed* has the entry (44b) as an alternative to the one given in (22). Then the PP f-structure serves the TO function, as shown in (45).

(44)
a. to: P, (↑ PCASE) = TO
b. handed: V, (↑ TENSE) = PAST
(↑ PRED) = 'HAND⟨(↑ SUBJ) (↑ OBJ) (↑ TO OBJ)⟩'

(45)

$$\left[\begin{array}{ll}
\text{SUBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`GIRL'}\end{array}\right]\\
\text{TENSE} & \text{PAST}\\
\text{PRED} & \text{`HAND}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ})\ (\uparrow \text{TO OBJ})\rangle\text{'}\\
\text{OBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`TOY'}\end{array}\right]\\
\text{TO} & \left[\begin{array}{ll}\text{PCASE} & \text{TO}\\ \text{OBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{THE}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`BABY'}\end{array}\right]\end{array}\right]
\end{array}\right]$$

The BABY f-structure is accessible as the TO OBJ, and it is correctly mapped onto the goal argument of HAND by the semantic form for *handed* in (44b) and (45). As mentioned earlier, this is systematically related to the semantic form in (22) by a dative lexical redundancy rule, so that the generalization marking sentences (2) and (41) as paraphrases is not lost.

Most of the statements in the f-description for (41) are either the same as or very similar to the statements in (25)–(27). The statements most relevant to the issue at hand are instantiated inside the prepositional phrase and at the PP node in the verb phrase:

(46)

a. $(f_3\ (f_5\ \text{PCASE})) = f_5$ from PP in (43a)

b. $(f_5\ \text{PCASE}) = \text{TO}$ from *to*

The designator on the left side of (46a) is of course the crucial one. This is processed by first locating the values of $f_3$ and $(f_5\ \text{PCASE})$, and then applying the first of these values to the second. If (46b) is processed before (46a), then the value of $(f_5\ \text{PCASE})$ will be the symbol TO, and (46a) will thus receive the same treatment as the more restricted equations we considered above.

We cannot insist that the f-description be processed in this or any other order, however. Since equality is an equivalence relation, whether or not an f-structure is a solution to a given f-description is not a property of any ordering on the f-description statements. An order dependency in our algorithm would simply be an artifact of its operation. Unless we could prove that an acceptable order can be determined for any set of statements, we would run the risk of ordering paradoxes whereby our algorithm does not produce a solution even though satisfactory f-structures do exist. A potential order dependency arises only when one equation establishes relationships between entities that have not yet been defined. Place-holders serve in our algorithm as temporary surrogates for those unknown entities. Our examples above illustrate their use in representing simple relationships. Changing the order in which equations (46a,b) are processed demonstrates that the proper treatment of more complicated cooccurrence relationships does not depend on a particular sequence of statements.

Suppose that (46a) is processed before (46b). Then the value of $(f_5\ \text{PCASE})$ will be a place-holder as shown in (47a), and $f_3$ will be assigned an f-structure with place-holders in both attribute and value positions, as in (47b):

(47)

a. $_{f_5}\left[\text{PCASE} \qquad \underline{\qquad}\right]$

b. $_{f_3}\left[\underline{\qquad} \qquad \underline{\qquad}\right]$

The value of the larger designator $(f_3\ (f_5\ \text{PCASE}))$ will thus be the second place-holder in (47b). When this is merged with the f-structure assigned to $f_5$, the result is (48):

(48)

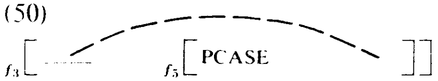
$$_{f_3}\left[\underline{\qquad} \qquad _{f_5}\left[\text{PCASE} \qquad \underline{\qquad}\right]\right]$$

It is not clear from (48) that the two blank lines stand for the same place-holder. One way of indicating this fact is to annotate blank lines with an identifying index whenever they represent occurrences of the same place-holder in multiple contexts, as shown in (49). An alternative and perhaps more perspicuous way of marking the important formal relationships is to display the blank line in just one of the place-holder's positions and then draw connecting lines to its other occurrences, as in (50):

(49)

$${}_{f_3}\left[\quad {}_{1} \qquad {}_{f_5}\left[\text{PCASE} \qquad {}_{1}\right]\right]$$

(50)



This problem of representation arises because our hierarchical f-structures are in fact directed graphs, not trees, so all the connections cannot easily be displayed in textual form. With the cooccurrences explicitly represented, processing equation (46b) causes the symbol TO to be substituted for the place-holder in both positions:

(51)

$${}_{f_3}\left[\text{TO} \qquad {}_{f_5}\left[\text{PCASE} \qquad \text{TO}\right]\right]$$

The index or connecting line is no longer needed, because the common spelling of symbols in two positions suffices to indicate their formal identity. The structure (51) is combined with the result of processing the remaining equations in the f-description, yielding the final structure (45).

The Kleene-star operator on the PP in (43a) allows for sentences having more than one oblique object:

(52)
The toy was given to the baby by the girl.

The f-structure of this sentence will have both a TO OBJ and a BY OBJ.Because of the functional well-formedness conditions discussed in the next section, these grammatical relations are compatible only with a semantic form that results from the passive lexical rule:

(53)
'HAND⟨(↑ BY OBJ) (↑ SUBJ) (↑ TO OBJ)⟩'

Although the c-structure rule suggests that any number of oblique objects are possible, they are in fact strictly limited by semantic form specifications. Moreover, if two prepositional phrases have the same preposition and hence the same PCASE feature, the Uniqueness Condition implies that only one of them can serve as an argument. If the sentence is to be grammatical, the other must be interpreted as some sort of adjunct. In (54), either *the policeman* or *the boy* must be a nonargument locative:

(54)
The baby was found by the boy by the policeman.

Thus, the PP element in rule (43a) derives the PP nodes for dative *to*-phrases, agentive *by*-phrases, and other, more idiosyncratic English oblique objects. Schemata similar to the one on the PP will be much more common in languages that make extensive use of lexically as opposed to structurally induced grammatical relations (e.g., heavily case-marked, nonconfigurational languages).

We have illustrated how our algorithm builds the f-structure for two grammatical sentences. However, as indicated above, f-descriptions which contradict the Uniqueness Condition are not solvable, and our algorithm must also inform us of this inconsistency. Consistency checking is carried out by both the Locate and the Merge operators. The Locate operator, for example, cannot succeed if a statement specifies that a symbol or semantic form is to be applied as a function or if a function is to be applied to an f-structure or semantic form argument. The string is marked ungrammatical if this happens. Similarly, a merger cannot be completed if the two entities to be merged are incompatible, either because they are of different types (a symbol and an f-structure, for example) or because they are otherwise in conflict (two distinct symbols or semantic forms, or two f-structures that assign distinct values to the same argument). Again, this means that the f-description is inconsistent.

Our algorithm thus produces *one* solution for an arbitrary consistent f-description, but it is not the *only* solution. If an f-structure *F* is a solution for a given f-description, then any f-structure formed from *F* by adding values for attributes not already present will also satisfy the f-description. Since the f-description does not mention those attributes or values, they cannot conflict with any of its statements. For example, we could add the arbitrary pairs X–Y and Z–W to the SUBJ f-structure of (40) to form (55):

(55)

$$\begin{bmatrix} \text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{'GIRL'} \\ \text{X} & \text{Y} \\ \text{Z} & \text{W} \end{bmatrix}$$

Substituting this for the original SUBJ value yields another solution for (25)–(27). This addition procedure, which defines a partial ordering on the set of f-structures, can be repeated indefinitely. In general, if an f-description has one solution, it has an infinite number of "larger" solutions.

Of course, there is something counterintuitive about these larger solutions. The extra features they contain cannot conflict with those specifically required by the f-description. In that sense they are grammatically irrelevant and should not really count as f-structures that the grammar assigns to sentences. This intuition, that we only countenance f-structures with relevant attributes and values, can be formalized in a technical refinement to our previous definitions that makes "*the* f-structure of a sentence" a well-defined notion.

Looking at the partial ordering from the opposite direction, an f-description may also have solutions smaller than a given one. These are formed by removing various combinations of its pairs (for example, removing the X–Y, Z–W pairs from (55) produces the smaller original solution in (40)). Some smaller f-structures are too small to be solutions of the f-description, in that they do not contain pairs that the f-description requires. For example, if the SPEC feature is removed from (55), the resulting structure will not satisfy equation (27a). We say that an f-structure $F$ is a *minimal* solution for an f-description if it meets all of the f-description's requirements and if no smaller f-structure also meets those requirements.

A minimal solution exists for every consistent f-description. By definition, each has at least one solution. Either that one is minimal, or there is a smaller solution. If that one is also not minimal, there is another, still smaller, solution. Since an f-structure has only a finite number of pairs to begin with, there are only a finite number of smaller f-structures. This sequence will therefore stop at a minimal solution after a finite number of steps.

However, the minimal solution of an f-description is not necessarily unique. The fact that f-structures are partially but not totally ordered means that there can be two distinct solutions to an f-description both of which are minimal but neither of which is smaller than the other. This would be the case for an f-description that contained the equation (56), asserting that the subject and object have the same person, if other equations were not included to specify that common feature's value.

(56)
(↑ SUBJ PERSON) = (↑ OBJ PERSON)

Any f-structure that is a minimal solution for all other equations of the f-description and contains any value at all for both the OBJ and SUBJ person features will also be a minimal solution for the larger f-description that includes (56). The values FIRST, SECOND, or THIRD, for instance, would all satisfy (56), but an f-structure without *some* person value would not be a solution. An f-description that does not have a unique minimal solution is called *indeterminate*. In effect, such an f-description does not have enough independent specifications for the number of unknown entities that it mentions.

We can now formulate a precise condition on the well-formedness of a string:

(57)
*Condition on Grammaticality*
A string is grammatical only if it has a valid c-structure with an associated f-description that is both consistent and determinate. The f-structure assigned to the string is the value in the f-description's unique minimal solution of the ↓-variable of the c-structure's root node. This condition is necessary but not sufficient for grammaticality; we later postulate additional requirements. As presented above, our solution algorithm decides whether or not the f-description is consistent and, if it is, constructs one solution for it. We observe that if no place-holders remain in that solution, it is the unique minimal solution: if any attribute or value is changed or removed, the resulting structure is not a solution since it no longer satisfies the equation the processing of which gave rise to that attribute or value. On the other hand, if there are residual place-holders in the f-structure produced by the algorithm, the f-description is indeterminate. Those place-holders can be replaced by any number of values to yield minimal solutions. Our algorithm is thus a decision procedure for all the functional conditions on grammaticality specified in (57).

## 4.5 Functional Well-formedness

The functional well-formedness conditions of our theory cause strings with otherwise valid c-structures to be marked ungrammatical. Our functional component thus acts as a filter on the output of the c-structure component, but in a sense that is very different from the way surface structure filtering has been used in transformational theory (e.g., Chomsky and Lasnik 1977). We do not allow arbitrary predicates to be applied to the c-structure output. Rather, we expect that a substantive linguistic theory will make available a universal set of grammatical functions and features and indicate how these may be assigned to particular lexical items and particular c-structure configurations. The most important of our well-formedness conditions, the Uniqueness Condition,[13] merely ensures that these assignments for a particular sentence are globally consistent so that its f-structure exists. Other general well-formedness conditions, the Completeness and Coherence Conditions, guarantee that grammatical functions and lexical predicates appear in mutually compatible f-structure configurations.

Consider the string (58), which is ungrammatical because the numbers of the final determiner and noun disagree:

(58)
*A girl handed the baby a toys.

The only f-description difference between this and our previous example is that the lexical entry for *toys* produces the equation (59) instead of (27l):

(59)
$(f_5\ \text{NUM}) = \text{PL}$

A conflict between the lexical specifications for *a* and *toys* arises because their schemata are attached to daughters of the same NP node. Some of the properties of that node's f-structure are specified by the determiner's lexical schemata and some by the noun's. According to the Uniqueness Condition, all properties attributed to it must be compatible if that f-structure is to exist. In the solution process for (58), $f_5$ will have the tentative value shown in (60) when equation (59) is encountered in place of (27l). The value of the lefthand designator is the symbol SG, which is incompatible with the PL value of the righthand designator. These two symbols cannot be merged.

(60)

$$f_5\begin{bmatrix} \text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \end{bmatrix}$$

The consistency requirement is a general mechanism for enforcing grammatical compatibilities among lexical items widely separated in

the c-structure. The items and features that will enter into an agreement are determined by both lexical and grammatical schemata. Number agreement for English subjects and verbs illustrates a compatibility that operates over a somewhat wider scope than agreement for determiners and nouns. It accounts for the unacceptability of (61):

(61)
*The girls hands the baby a toy.

The grammar fragment in (21) needs no further elaboration in order to reject this string. The identification on the VP in (21a) indicates that one f-structure corresponds to both the S and the VP nodes. This implies that any constraints imposed on a SUBJ function by the verb will in fact apply to the SUBJ of the sentence as a whole, the f-structure corresponding to the first NP. Thus, the following lexical entry for *hands* ensures that it will not cooccur with the plural subject *girls:*

(62)
hands: V, $(\uparrow \text{TENSE}) = \text{PRESENT}$
$(\uparrow \text{SUBJ NUM}) = \text{SG}$
$(\uparrow \text{PRED}) = \text{`HAND}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ2})\ (\uparrow \text{OBJ})\rangle\text{'}$

The middle schema, which is contributed by the present tense morpheme, specifies the number of the verb's subject. It is instantiated as (63a), and this is inconsistent with (63b), which would be derived from the lexical entry for *girls:*

(63)
a. $(f_3\ \text{SUBJ NUM}) = \text{SG}$
b. $(f_2\ \text{NUM}) = \text{PL}$

The conflict emerges because $f_2$ is the SUBJ of $f_1$, and $f_1$ is equal to $f_3$.

We rely on violations of the Uniqueness Condition to enforce many cooccurrence restrictions besides those that are normally thought of as agreements. For example, the restrictions among the elements in an English auxiliary sequence can be handled in this way, even though the matching of features does not at first seem to be involved. There is a natural way of coding the lexical features of auxiliaries, participles, and tensed verbs so that the "affix-hopping" phenomena follow as a consequence of the consistency requirement. Auxiliaries can be treated as main verbs that take embedded VP′ complements. We expand our grammar as shown in (64) in order to derive the appropriate c-structures. (The optional *to* permitted by rule (64b), while necessary for other types of VP complements, does not appear with most auxiliary heads. This restriction could be imposed by an additional schema.)

(64)

a. $$\text{VP} \rightarrow \text{V} \left(\underset{(\uparrow\ \text{OBJ})=\downarrow}{\text{NP}}\right) \left(\underset{(\uparrow\ \text{OBJ2})=\downarrow}{\text{NP}}\right) \underset{(\uparrow\ (\downarrow\ \text{PCASE}))=\downarrow}{\text{PP*}} \left(\underset{(\uparrow\ \text{VCOMP})=\downarrow}{\text{VP}'}\right)$$

b. $$\text{VP}' \rightarrow (\textit{to})\ \underset{\uparrow=\downarrow}{\text{VP}}$$

Rule (64a) allows an optional VP′ following the other VP constituents. Of course, auxiliaries exclude all the VP possibilities except the VCOMP; this is enforced by general completeness and coherence conventions, as described below. For the moment, we focus on their affix cooccurrence restrictions, which are represented by schemata in the lexical entries for verbs. Each nonfinite verb will have a schema indicating that it is an infinitive or a participle of a particular type, and each auxiliary will have an equation stipulating the inflectional form of its VCOMP. (A small number of additional features are needed to account for the finer details of auxiliary ordering and for other cooccurrence restrictions, as noted for example by Akmajian, Steele, and Wasow 1979.) The lexical entries in (65)–(66) are for *handing* considered as a present participle (as opposed to a past tense or passive participle form) and for *is* as a progressive auxiliary:[14]

(65)
handing: V, (↑ PARTICIPLE) = PRESENT
(↑ PRED) = 'HAND⟨(↑ SUBJ) (↑ OBJ2) (↑ OBJ)⟩'

(66)
is: V,
a. (↑ TENSE) = PRESENT
b. (↑ SUBJ NUM) = SG
c. (↑ PRED) = 'PROG⟨(↑ VCOMP)⟩'
d. (↑ VCOMP PARTICIPLE) = PRESENT
e. (↑ VCOMP SUBJ) = (↑ SUBJ)

Schema (66d) stipulates that the PARTICIPLE feature of the verb phrase complement must have the value PRESENT. The VCOMP is defined in (64a) as the ↓ f-structure of the VP′ node, and this is identified with the ↓ f-structure of the VP node by the schema in (64b). This means that the PARTICIPLE stipulations for *handing* and *is* both hold of the same f-structure. Hence, sentence (67a) is accepted but (67b) is rejected because *has* demands of its VCOMP a non-PRESENT participle:

(67)
a. A girl is handing the baby a toy.
b. *A girl has handing the baby a toy.

Schemata (66c,e) deserve special comment. The semantic form for *is* specifies that the logical formula derived by interpreting the VCOMP function is the single argument of a predicate for progressiveness. Even though the f-structure for (67a) will include a SUBJ function at the level of the PROG predicate, that function does not serve as an argument of PROG. Instead, it is asserted by (66e) to be equivalent to the SUBJ at the *handing* level. This would not otherwise exist, because there is no subject NP in the VP′ expansion. The effect is that the *girl* is correctly interpreted as the first argument of HAND. (66e) is an example of a schema for *functional control*, which we will discuss more fully below.

These illustrations of the filtering effect of the Uniqueness Condition have glossed over an important conceptual distinction. A schema is often included in a lexical entry or grammatical rule in order to *define* the value of some feature. That is, instantiations of that schema provide sufficient grounds for inserting the feature–value pair into the appropriate f-structure (assuming of course that there is no conflict with the value defined by other equations). However, sometimes the purpose of a schema is only to *constrain* a feature whose value is expected to be defined by a separate specification. The feature remains valueless when the f-description lacks that specification. Intuitively, the constraint is not satisfied in that case and the string is to be excluded. Constraints of this sort thus impose stronger well-formedness requirements than the definitional inconsistency discussed above.

Let us reexamine the restriction that schema (66d) imposes on the participle of the VCOMP of *is*. We have seen how this schema conspires with the lexical entries for *handing* (65) and *has* to account for the facts in (67). Intuitively, it seems that the same present participle restriction ought to account for the unacceptability of (68):

(68)
*A girl is hands the baby a toy.

This string will not be rejected, however, if *hands* has the lexical entry in (62) and (66d) is interpreted as a defining schema. The PARTICIPLE feature has no natural value for the finite verb *hands*, and (62) therefore has no specification at all for this feature. This permits (66d) to define the value PRESENT for that feature without risk of inconsistency, and the final f-structure corresponding to the *hands* VP will actually contain a PARTICIPLE–PRESENT pair. We have concluded that *hands* is a present participle just because *is* would like it to be that way! If, on the other hand, we interpret (66d) as a constraining schema, we are prevented from making this implausible inference and the string is appropriately rejected. The constraining interpretation is clearly preferable.

Introducing a special interpretation for f-description statements is not strictly necessary to account for these facts. We could allow only the defining interpretation of equations and still obtain the right pattern of results by means of additional feature specifications. For example, we could insist that there be a PARTICIPLE feature for every verbal form, even finite forms that are notionally not participles at all. The value for tensed forms might be NONE, and this would be distinct from and thus conflict with PRESENT and all other real values. The lexical entry for *hands* would become (69), and (68) would be ruled out even with a defining interpretation for (66d):

(69)
hands: V, (↑ PARTICIPLE) = NONE
(↑ TENSE) = PRESENT
(↑ SUBJ NUM) = SG
(↑ PRED) = 'HAND⟨(↑ SUBJ) (↑ OBJ2) (↑ OBJ)⟩'

There are two objections to the presence of such otherwise unmotivated features: they make the formal system more cumbersome for linguists to work with and less plausible as a characterization of the linguistic generalizations that children acquire. Lexical redundancy rules in the form of marking conventions provide a partial answer to both objections. A redundancy rule, for example, could assign special no-value schemata to every lexical entry that is not already marked for certain syntactic features. Then the NONE schema would not appear in the entry for *hands* but would still be available for consistency checking.

Although we utilize lexical redundancy rules to express a variety of other generalizations, we have chosen an explicit notational device to highlight the conceptual distinction between definitions and constraints. The ordinary equal-sign that has appeared in all previous examples indicates that a schema is definitional, while an equal-sign with the letter *c* as a subscript indicates that a schema expresses a constraint. With this notation, the lexical entry for *is* can be formulated more properly as (70):

(70)
is: V, (↑ TENSE) = PRESENT
(↑ SUBJ NUM) = SG
(↑ PRED) = 'PROG⟨(↑ VCOMP)⟩'
(↑ VCOMP PARTICIPLE) $=_c$ PRESENT
(↑ VCOMP SUBJ) = (↑ SUBJ)

The notational distinction is preserved when the schemata are instantiated, so that the statements in an f-description are also divided into two classes. Defining equations are interpreted by our solution algorithm in the manner outlined above and thus provide evidence for actually constructing satisfactory structures. Constraining equations are simply not given to the solution algorithm. They are reserved until all defining equations have been processed and all variables have been assigned final f-structure values. At that point, the constraining equations are evaluated, and the string is accepted only if they all turn out to be true. This difference in interpretation accurately reflects the conceptual distinction represented by the two types of equations. It also gives the right result for string (68): since the revised VCOMP requirement in (70) will be false for the f-structure constructed from its defining equations, that string will be rejected without adding the special NONE value to *hands*.

Whether or not a particular cooccurrence restriction should be enforced by consistency among defining equations or the later evaluation of constraining equations depends on the meaning that is most naturally assigned to the absence of a feature specification. A constraining equation is appropriate if, as in the examples above, an unspecified value is intended to be in conflict with all of a feature's real values. On the other hand, a value specification may be omitted for some features as an indication of vagueness, and the restriction is then naturally stated in terms of a defining equation.[15] The case features of English nouns seem to fall into this second category: only pronouns have explicit nominative/accusative markings; all other nouns are intuitively unmarked, yet may appear in either subject or object positions. The new subject-NP schema in (71) defines the subject's case to be NOM. The NOM value will thus be included in the f-structure for any sentence with a nominative pronoun or nonpronoun subject. Only strings with accusative pronouns in subject position will have inconsistent f-descriptions and be excluded.

(71)

$$\text{S} \rightarrow \underset{\substack{(\uparrow \text{SUBJ})=\downarrow \\ (\downarrow \text{CASE})=\text{NOM}}}{\text{NP}} \qquad \underset{\substack{\uparrow=\downarrow \\ (\uparrow \text{TENSE})}}{\text{VP}}$$

Defining schemata always assert particular values for features and thus always take the form of equations. For constraints, two nonequational specification formats also make sense. The new TENSE schema in (71), for example, is just a designator not embedded in an equality. An instantiation of such a constraint is satisfied just in case the expression has *some* value in the final f-structure; these are called *existential* constraints. The TENSE schema thus expresses the requirement that S-clauses must have tensed verbs and rules out strings like (72):

(72)
*A girl handing the baby a toy.

As with equational constraints, it is possible to achieve the effect of an existential schema by introducing ad hoc feature values (e.g., one that discriminates tensed forms from all other verbals), but this special constraint format more directly represents the intuitive content of the requirement.

Finally, constraints may also be formed by adding a negation operator to an equational or existential constraint. The sentence is then acceptable only if the constraint without the negation turns out to be false. Such constraints fall quite naturally within our formal framework and may simplify a variety of grammatical descriptions. The negative existential constraint in (73), for example, is one way of stipulating that the VP after the particle *to* in a VP′ is untensed:

(73)

$$\text{VP}' \rightarrow \begin{pmatrix} \textit{to} \\ \neg\,(\uparrow \text{TENSE}) \end{pmatrix} \quad \underset{\uparrow=\downarrow}{\text{VP}}$$

According to these well-formedness conditions, strings are rejected when an f-structure cannot be found that simultaneously satisfies all the explicit defining and constraining statements in the f-description. LFG also includes implicit conventions whose purpose is to make sure that f-structures contain mutually compatible combinations of lexical predicates and grammatical functions. These conventions are defined in terms of a proper subset of all the features and functions that may be represented in an f-structure. That subset consists of all functions

whose values can serve as arguments to semantic predicates,[16] such as subject and various objects and complements. We refer to these as the *governable grammatical functions*. A given lexical entry mentions only a few of the governable functions, and we say that that entry *governs* the ones it mentions. (For a fuller discussion of government in lexical-functional theory, see chapter 5 of this volume.) Our conditions of functional compatibility simply require that an f-structure contain all of the governable functions that the lexical entry of its predicate actually governs, and that it contain no other governable functions.

This compatibility requirement gives a natural account for many types of ill-formedness. The English c-structure grammar, for example, must permit verbs not followed by NP arguments so that ordinary intransitive sentences can be generated. However, the necessary intransitive VP rule can then be applied with a verb that normally requires objects to yield a c-structure and f-structure for ill-formed strings such as (74):

(74)
*The girl handed.

The unacceptability of this string follows from the fact that the lexical entry for *handed* governs the grammatical functions OBJ and OBJ2 or TO OBJ, which do not appear in its f-structure. On the other hand, there is nothing to stop the c-structure rule that generates objects from applying in strings such as (75), where the verb is intransitive.

(75)
*The girl fell the apple the dog.

This string exhibits the opposite kind of incompatibility: the governable functions OBJ and OBJ2 do appear in its f-structure but are not governed by the intransitive verb *fell*.

Stated in more technical terms, string (74) is ungrammatical because its f-structure is not *complete*, while (75) fails because its f-structure is not *coherent*. These properties of f-structures are precisely defined as follows:

(76)
*Definitions of Completeness and Coherence*

a. An f-structure is *locally complete* if and only if it contains all the governable grammatical functions that its predicate governs. An f-structure is *complete* if and only if it and all its subsidiary f-structures are locally complete.

b. An f-structure is *locally coherent* if an only if all the governable grammatical functions that it contains are governed by a local predicate. An f-structure is *coherent* if and only if it and all its subsidiary f-structures are locally coherent.

Functional compatibility then enters into our notion of grammaticality by way of the following obvious condition:

(77)
*Grammaticality Condition*
A string is grammatical only if it is assigned a complete and coherent f-structure.

Since coherence and completeness are defined in terms of local configurations of functions, there are straightforward ways of formally verifying that these conditions are satisfied. For example, a set of constraints that encode these requirements can be added to all f-descriptions by a simple redundancy convention. We identify a set of *governable designators* corresponding to the governable grammatical functions and a set of *governed designators* corresponding to the functions governed by a particular lexical entry. The set of governable designators for a language is simply a list of every designator that appears as an argument in a semantic form for at least one entry in the lexicon. Thus, the set of governable designators for English includes (↑ SUBJ), (↑ OBJ), (↑ BY OBJ), (↑ VCOMP), etc. The set of governed designators for a particular lexical entry then contains only those members of the governable list that appear in that entry. If existential constraints for all the governed designators are instantiated along with the other schemata in the lexical entry, then the f-structure in which the lexical predicate appears will be locally complete if and only if it satisfies all those constraints. The f-structure will be locally coherent if and only if *negative* existential constraints for all the governable but ungoverned designators are also satisfied. Under this interpretation, example (74) above is incomplete because its f-structure does not satisfy the constraining schema (↑ OBJ) and (75) is incoherent because ¬ (↑ OBJ) is not satisfied.

It is important to observe that a designator is considered to be governed by an entry if it appears anywhere in the entry, not solely in the semantic form argument-list (though to be governable, it must appear as an argument in *some* lexical entry). In particular, the designator may appear only in a functional control schema or only in a schema defining or constraining some feature. Thus, the lexical entry for *is* in (66) above is considered to govern the designator (↑ SUBJ) because of its appearance in both the number-defining schema and the control schema for the VCOMP's SUBJ. (↑ SUBJ), however, is not assigned to an argument in the semantic form 'PROG⟨(↑ VCOMP)⟩'.

A grammatical function is also considered to be governed by an entry even when its value is constrained to be a semantically empty syntactic formative. Among these formatives are the expletives *there* and *it*, plus the components of various idiomatic expressions (e.g., the idiomatic sense of *tabs* in the expression *keep tabs on*). The lexicon marks such items as being in ordinary syntactic categories (pronoun or noun, for example), but their schemata specify a symbol value for a FORM attribute instead of a semantic form value for a PRED attribute:

(78)
tabs: N, (↑ FORM) = TABS
(↑ NUM) = PL

A *tabs* NP may appear in any c-structure NP position and will be assigned the associated grammatical function. The Coherence Condition ensures that that function is governed by the lexical head of the f-structure; (79) is ruled out for the same reason that (75) is ill-formed:

(79)
*The girl fell tabs.

If the f-structure is coherent, then its lexical head makes some specification about the *tabs* function. For the acceptable sentence (80), the lexical entry for the idiomatic *kept* has a constraining schema for the necessary FORM value, as illustrated in (81):

(80)
The girl kept tabs on the baby.

(81)
kept: V, (↑ TENSE) = PAST
(↑ PRED) = 'OBSERVE⟨(↑ SUBJ) (↑ ON OBJ)⟩'
(↑ OBJ FORM) $=_c$ TABS

This constraining schema precludes the OBSERVE reading of *kept* with the nonidiomatic OBJ in (82a) and also rejects OBJS with the wrong formative feature (82b):

(82)
a. *The girl kept the dog on a baby.
b. *The girl kept there on a baby.

The ill-formedness of (83), however, is not predicted from the functional compatibility conditions we have presented:

(83)
*The girl handed there tabs.

In this example a governed function serving as an argument to the predicate HAND has a semantically empty value. A separate condition of semantic completeness could easily be added to our grammaticality requirements, but such a restriction would be imposed independently by a semantic translation procedure. A separate syntactic stipulation is therefore unnecessary.

In this section we have described several mechanisms for rejecting as functionally deviant strings that have otherwise valid c-structure derivations. The Uniqueness Condition is the most basic well-formedness requirement, since an f-structure does not even exist if it is not satisfied. If an f-structure does exist, it must satisfy any constraining schemata and the Completeness and Coherence Conditions must hold. The combined effect of these conventions is to impose very strong restrictions among the components of a sentence's f-structure and c-structure, so that semantic forms and grammatical formatives can appear only in the appropriate functional and constituent environments. Because of these functional well-formedness conditions, there is no need for a separate notion of c-structure subcategorization to guarantee that lexical cooccurrence restrictions are satisfied. Indeed, Grimshaw in preparation and Maling 1980 suggest that an account of lexical cooccurrences based on functional compatibility is superior to one based on subcategorization.

These mechanisms ensure that syntactic compatibility holds between a predicate and its arguments. A sentence may have other elements, however, that are syntactically related to the predicate but are not syntactically restricted by it. These are the adverbial and prepositional modifiers that serve as *adjuncts* of a predicate. Although adjuncts and predicates must be associated in an f-structure so that the correct semantic relationship can be determined, adjuncts are not within range of a predicate's syntactic schemata. A predicate imposes neither category nor feature restrictions on its adjuncts, semantic appropriateness being the only requirement that must be satisfied. As the temporal adjuncts in sentence (84) illustrate, adjuncts do not even obey the Uniqueness Condition.

(84)
The girl handed the baby a toy on Tuesday in the morning.

Since adjuncts do not serve as arguments to lexical predicates, they are not governable functions and are thus also immune to the Completeness and Coherence Conditions.

Given the formal devices we have so far presented, there is no f-structure representation of adjuncts that naturally accounts for these properties. If an individual adjunct is assigned as the value of an attribute (e.g., TEMP, LOC, or simply ADJUNCT), the Uniqueness Condition is immediately applicable and syntactic cooccurrence restrictions can in principle be stated. However, the shared properties of adjuncts do follow quite naturally from a simple extension to the notion of what a possible value is. Besides the individual f-structure values for the basic grammatical relations, we allow the value of an attribute to be a *set* of f-structures. Values of this type are specified by a new kind of schema in which the membership symbol ∈ appears instead of a defining or constraining equal-sign.

The membership schema $\downarrow \in (\uparrow \text{ADJUNCTS})$ in the VP rule (85), for example, indicates that the value of ADJUNCTS is a set containing the PP's f-structure as one of its elements.

(85)

$$\begin{array}{llll} \text{VP} \rightarrow \text{V} \quad \text{NP} & \text{NP} & \text{PP}^* \\ \quad (\uparrow \text{OBJ})=\downarrow & (\uparrow \text{OBJ2})=\downarrow & \downarrow\in(\uparrow \text{ADJUNCTS}) \end{array}$$

The * permits any number of adjuncts to be generated, and the ↓ metavariable will be instantiated differently for each one. The f-description for sentence (84) will thus have two membership statements, one for the *on Tuesday* PP and one for *in the morning*. These statements will be true only of an f-structure in which ADJUNCTS has a set value containing one element that satisfies all other statements associated with *on Tuesday* and another element satisfying the other statements of *in the morning*. The outline of such an f-structure is shown in (86):

(86)

$$\left[\begin{array}{ll}
\text{SUBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`GIRL'}\end{array}\right]\\
\text{TENSE} & \text{PAST}\\
\text{PRED} & \text{`HAND}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ2})\ (\uparrow \text{OBJ})\rangle\text{'}\\
\text{OBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{THE}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`BABY'}\end{array}\right]\\
\text{OBJ2} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`TOY'}\end{array}\right]\\
\text{ADJUNCTS} & \{\text{``on Tuesday'' ``in the morning''}\}
\end{array}\right]$$

The braces in this representation surround the elements of the set value; they are distinct from the braces in c-structure rules that indicate alternative expansions. We have elided the adjuncts' internal functions since they are not immediately relevant to the issue at hand and are the topic of current syntactic and semantic research (e.g., chapter 6 of this volume and Halvorsen in preparation).

The peculiar properties of adjuncts now follow from the fact that they are treated syntactically as elements of sets. Membership statements define adjuncts to be elements of a predicate's adjunct "pool," but there is no requirement of mutual syntactic compatibility among the various elements. Hence, the Uniqueness Condition does not apply. Further, since there is no notation for subsequently referring to particular members of that set, there is no way that adjuncts can be restricted by lexical schemata associated with the predicate.[17] Adjuncts are susceptible only to conditions that can be stated on the rule elements that generate them. Their category can be specified, and feature requirements can be imposed by schemata involving the ↓ metavariable. Since reference to the adjunct via ↓ is not possible from other places in the string, our formal system makes adjuncts naturally context-free. (Conjoined elements are similar to adjuncts in some of these respects and might also be represented in an f-structure as sets.)

Although the PP in (85) appears in the same position as the oblique object PP category in our previous VP rule, the schemata on the two PP rule elements are quite different and apply to alternative lexical entries of the preposition. The oblique object requires the case-marking lexical entry (with the PCASE feature defined), while semantic translation of the adjunct requires the predicate alternative of the preposition. Adjuncts and oblique objects can both appear in the same sentence and in any order, as illustrated by (87a,b),[18] and sometimes a PP may be interpreted ambiguously as either an adjunct or an oblique object, as in (87c):

(87)

a. The baby was handed the toy at five o'clock by the girl.
b. The baby was handed the toy by the girl at five o'clock.
c. The baby was handed the toy by the girl by the policeman.

To account for these facts, the adjunct possibility must be added as an alternative to the oblique object PP in our previous VP rule (64a). The star operator outside the braces in (88) means that the choice between the two PPs may be repeated arbitrarily.

(88)

$$\text{VP} \rightarrow \text{V} \left(\begin{array}{c}\text{NP}\\ (\uparrow \text{OBJ})=\downarrow\end{array}\right) \left(\begin{array}{c}\text{NP}\\ (\uparrow \text{OBJ2})=\downarrow\end{array}\right) \left\{\begin{array}{c}\text{PP}\\ (\uparrow (\downarrow \text{PCASE}))=\downarrow\\ \\ \text{PP}\\ \downarrow\in(\uparrow \text{ADJUNCTS})\end{array}\right\}^{*} \left(\begin{array}{c}\text{VP}'\\ (\uparrow \text{VCOMP})=\downarrow\end{array}\right)$$

An equivalent but more compact formulation of this rule is given in (89). We have factored the common elements of the two PP alternatives, moving the braces so that they enclose just the alternative schemata.

(89)

$$\text{VP} \rightarrow \text{V} \left(\begin{array}{c}\text{NP}\\ (\uparrow \text{OBJ})=\downarrow\end{array}\right) \left(\begin{array}{c}\text{NP}\\ (\uparrow \text{OBJ2})=\downarrow\end{array}\right) \begin{array}{c}\text{PP*}\\ \left\{\begin{array}{c}(\uparrow (\downarrow \text{PCASE}))=\downarrow\\ \downarrow\in(\uparrow \text{ADJUNCTS})\end{array}\right\}\end{array} \left(\begin{array}{c}\text{VP}'\\ (\uparrow \text{VCOMP})=\downarrow\end{array}\right)$$

A simple extension to our solution algorithm permits the correct interpretation of membership statements. We use a new operator *Include* for membership statements, just as we use *Merge* for equalities. If $d_1$ and $d_2$ are designators, a statement of the form $d_1 \in d_2$ is processed by performing Include[Locate[$d_1$], Locate[$d_2$]]. As formally defined in the appendix, the Include operator makes the value located for the first designator be an element of the set value located for the second designator; the f-description is marked inconsistent if that second value is known not to be a set. With this extension, our algorithm becomes a decision procedure for f-descriptions that contain both membership and equality statements.

### 4.6 Levels of Representation

We have now covered almost all the major structures and mechanisms of lexical-functional grammar, except for the bounded tree relations that govern long-distance grammatical dependencies. We postpone that discussion for still a few more pages in order to first review and reinforce some earlier claims.

We said at the outset that constituent structures and functional structures are formally quite different, and the descriptions of the preceding pages have amplified that point considerably. However, the mechanisms of our formal system—the immediate domination metavariables and the various grammatical and lexical schemata—presuppose and also help to establish a very close, systematic connection between the two levels of representation. Our claim of formal distinctness would of course be meaningless if this close connection turned out to be an isomorphism, so it is worth describing and motivating some ways in which c-structures and f-structures for English diverge. We show that individual c-structure nodes are not isomorphic to subsidiary f-structures for particular sentences and, more generally, that there is no simple relationship between node configurations and grammatical functions.

We observe first that our instantiation procedure defines only a partial correspondence between c-structure nodes and subsidiary f-structures. There are both c-structure nodes with no corresponding f-structures and also f-structures that do not correspond to c-structure nodes. The former situation is illustrated in our previous examples by every c-structure node which is not assigned a $\downarrow$-variable and therefore has no $\downarrow$ f-structure. The English imperative construction gives a simple illustration of the latter case: the subsidiary f-structure representing 'YOU' as the "understood" subject is not associated with a c-structure node. Plausible c- and f-structures for the imperative sentence (90a) would be generated by the alternative expansion for S in (90b), assuming that the lexical entry for *hand* has a +-valued INF(initive) feature. (A more realistic example would specify an imperative mood marker and perhaps other features.)

(90)

a. Hand the baby a toy.

b. $$S \rightarrow \begin{array}{c} \text{VP} \\ \uparrow=\downarrow \\ (\uparrow \text{INF})=_c+ \\ (\uparrow \text{SUBJ PRED})=\text{`YOU'} \end{array}$$

With this rule, the c-structure contains no NP dominated by S, yet the $\downarrow$ f-structure of the S node has as its SUBJ another full-fledged f-structure, defined completely by grammatical schemata:

(91)

$$\left[\begin{array}{ll} \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{`YOU'} \end{array}\right] \\ \text{INF} & + \\ \text{PRED} & \text{`HAND}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ2})\ (\uparrow \text{OBJ})\rangle\text{'} \\ \text{OBJ} & \left[\begin{array}{ll} \text{SPEC} & \text{THE} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`BABY'} \end{array}\right] \\ \text{OBJ2} & \left[\begin{array}{ll} \text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`TOY'} \end{array}\right] \end{array}\right]$$

A standard transformational grammar provides a dummy NP as a deep structure subject so that the correct semantic interpretation can be constructed and the necessary cooccurrence restrictions enforced. Our functional subject is sufficient for these purposes; the dummy NP is without surface justification and therefore does not appear in the c-structure.

Second, when nodes and subsidiary f-structures do correspond, the correspondence is not necessarily one-to-one. An identification schema, for example, usually indicates that two distinct nodes are mapped onto a single f-structure. In (40) a single f-structure is assigned to the $\downarrow$-variables for both the S and VP nodes in the c-structure (24), in accordance with the identification equation (25b). The two distinct nodes exist in (24) to capture certain generalizations about phrase structure cooccurrences and phonological patterns. The identification has the effect of "promoting" the functional information associated with the VP so that it is at the same hierarchical level as the SUBJ. This brings the SUBJ within range of the PRED semantic form, simplifying the statement of the Completeness and Coherence Conditions and allowing a uniform treatment of subjects and objects. As noted above, this kind of promotion also permits lexical specification of certain contextual restrictions, such as subject–verb number agreements.

Let us now consider the relationship between configurations of c-structure nodes and grammatical functions. The imperative example shows that a single functional role can be filled from distinct node configurations. While it is true for English that an S-dominated NP always yields a SUBJ function, a SUBJ can come from other sources as well. The grammatical schema on the VP for the imperative actually defines the SUBJ's semantic form. For a large class of other examples, the understood subject (that is, not from an S–NP configuration) is supplied through a schema of *functional control*. Control schemata, which identify grammatical relations at two different levels in the f-structure hierarchy, offer a natural account for so-called "equi" and "raising" phenomena.[19]

Sentence (92) contains the equi-type verb *persuaded*. The intuitive interpretation of the *baby* NP in this sentence is as an argument of both PERSUADE and GO. This interpretation will be assigned if *persuaded* has the lexical entry (93), given our previous VP rule (88) and the new schemata in (94) for the VP′'s optional *to*.

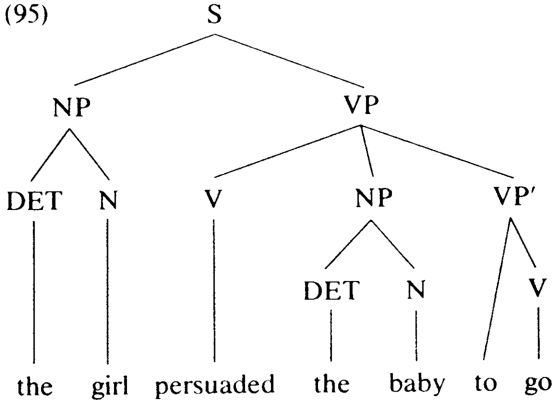
(92)
The girl persuaded the baby to go.

(93)
persuaded: V, (↑ TENSE) = PAST
(↑ PRED) = 'PERSUADE⟨(↑ SUBJ) (↑ OBJ) (↑ VCOMP)⟩'
(↑ VCOMP TO) $=_c$ +
(↑ VCOMP SUBJ) = (↑ OBJ)

(94)
$$\mathrm{VP}' \rightarrow \begin{pmatrix} \textit{to} \\ (\uparrow \text{TO})=+ \\ (\uparrow \text{INF})=_c+ \end{pmatrix} \begin{array}{c} \mathrm{VP} \\ \uparrow=\downarrow \end{array}$$

Our rules generate a c-structure in which *persuaded* is followed by an NP and a VP′, where the VP′ is expanded as a *to*-complement. This is shown in (95):

(95)



The f-structure for the *baby* NP becomes the OBJ of *persuaded* and the VP′ provides the VCOMP. The control schema, the last one in (93), identifies the OBJ f-structure as being also the SUBJ of the VCOMP. That f-structure thus appears in two places in the functional hierarchy (96):

(96)

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{SPEC} & \text{A} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`GIRL'} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \\ \text{PRED} & \text{`PERSUADE}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ})\ (\uparrow \text{VCOMP})\rangle\text{'} \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{THE} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`BABY'} \end{bmatrix} \\ \text{VCOMP} & \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{SPEC} & \text{THE} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{`BABY'} \end{bmatrix} \\ \text{INF} & + \\ \text{TO} & + \\ \text{PRED} & \text{`GO}\langle(\uparrow \text{SUBJ})\rangle\text{'} \end{bmatrix} \end{bmatrix}$$

The complement in this f-structure has essentially the same grammatical relations that would be assigned to the *that*-complement sentence (97), even though the c-structure for the *that*-complement is quite different:

(97)
The girl persuaded the baby that the baby (should) go.

The contrast between oblique objects and adjuncts shows that similar c-structure configurations—a VP dominating a PP—can be mapped into distinct grammatical functions. A comparison of the equi verbs *persuaded* and *promised* provides another illustration of the same point. Sentence (98) is the result of substituting *promised* for *persuaded* in sentence (92):

(98)
The girl promised the baby to go.

This substitution does not change the c-structure configurations, but for (98) the *girl*, not the *baby*, is understood as an argument of both the matrix and complement predicates. This fact is easily accounted for if the control schema in the lexical entry for *promised* identifies the complement SUBJ with the matrix SUBJ instead of the matrix OBJ:

(99)

$$\begin{array}{ll}
\text{promised: V,} & (\uparrow \text{TENSE}) = \text{PAST}\\
 & (\uparrow \text{PRED}) = \text{`PROMISE}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ})\ (\uparrow \text{VCOMP})\rangle\text{'}\\
 & (\uparrow \text{VCOMP TO}) =_c +\\
 & (\uparrow \text{VCOMP SUBJ}) = (\uparrow \text{SUBJ})
\end{array}$$

With this lexical entry, the f-structure for (98) correctly defines GIRL as the argument of GO in (100). The f-structure difference for the two types of equi verbs thus follows from the differing functional control schemata in their lexical entries, not from any c-structure difference.

(100)

$$\left[\begin{array}{ll}
\text{SUBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`GIRL'}\end{array}\right]\\
\text{TENSE} & \text{PAST}\\
\text{PRED} & \text{`PROMISE}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ})\ (\uparrow \text{VCOMP})\rangle\text{'}\\
\text{OBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{THE}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`BABY'}\end{array}\right]\\
\text{VCOMP} & \left[\begin{array}{ll}
\text{SUBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`GIRL'}\end{array}\right]\\
\text{INF} & +\\
\text{TO} & +\\
\text{PRED} & \text{`GO}\langle(\uparrow \text{SUBJ})\rangle\text{'}
\end{array}\right]
\end{array}\right]$$

From a formal point of view, there is no restriction on which grammatical relations in the matrix and complement may be identified by a schema for functional control. Very strong limitations, however, are imposed by the substantive linguistic theory that is based on our lexical-functional formalism. As discussed in chapter 5, the functional control schemata of human languages universally identify the SUBJ of a complement with the SUBJ, OBJ, or OBJ2 of the matrix. (The TOPIC function in English relative clauses and in *tough*-movement constructions may also be functionally controlled, as described in section 4.7.) Control schemata for VP complements different from those above for *promised* and *persuaded* may not appear in the grammar or lexicon of any human language. This universal stipulation explains the familiar contrast in the passivization behavior of *persuade* and *promise:*

(101)
a. The baby was persuaded to go by the girl.
b. *The baby was promised to go by the girl.

In chapter 1 it was argued that the systematic relationship between actives and their corresponding passives can be expressed by a universal lexical rule. In simple terms, this rule asserts that for any language, if an active lexical entry for a stem mentions the SUBJ and OBJ functions, then there is a passive lexical entry based on the same stem in which SUBJ is replaced by an oblique-object function and OBJ is replaced by SUBJ. For English, the passive oblique object is marked by the preposition *by*, so the English instance of this universal rule is as follows. (See chapter 1 of this volume for a discussion of the morphological changes that go along with these functional replacements.)

(102)

$$\begin{array}{ll} (\uparrow \text{SUBJ}) \rightarrow (\uparrow \text{BY OBJ}) & \\ (\uparrow \text{OBJ}) \mapsto (\uparrow \text{SUBJ}) & (\uparrow \text{PARTICIPLE}) = \text{PASSIVE} \end{array}$$

This rule indicates the replacements to be performed and also specifies that a PARTICIPLE schema appears in passive entries in addition to other schemata derived from the stem. Accordingly, the passive lexical entries based on the stems underlying the past tense forms *persuaded* and *promised* are as follows:

(103)

a. persuaded: V, $(\uparrow \text{PARTICIPLE}) = \text{PASSIVE}$
$(\uparrow \text{PRED}) = \text{`PERSUADE}\langle(\uparrow \text{BY OBJ})\ (\uparrow \text{SUBJ})\ (\uparrow \text{VCOMP})\rangle\text{'}$
$(\uparrow \text{VCOMP TO}) =_c +$
$(\uparrow \text{VCOMP SUBJ}) = (\uparrow \text{SUBJ})$

b. promised: V, $(\uparrow \text{PARTICIPLE}) = \text{PASSIVE}$
$(\uparrow \text{PRED}) = \text{`PROMISE}\langle(\uparrow \text{BY OBJ})\ (\uparrow \text{SUBJ})\ (\uparrow \text{VCOMP})\rangle\text{'}$
$(\uparrow \text{VCOMP TO}) =_c +$
$(\uparrow \text{VCOMP SUBJ}) = (\uparrow \text{BY OBJ})$

Notice that ($\uparrow$ SUBJ) and ($\uparrow$ OBJ), the lefthand designators in the lexical rule, are replaced inside semantic forms as well as in schemata. The control schema in (103a) conforms to the universal restriction on functional control, but the one in (103b) does not. Since (103b) is not a possible lexical entry, *promise* may not be passivized when it takes a VP complement.

We have argued that the *to*-complement and *that*-complement of *persuaded* have essentially the same internal functions. The sentences (92) and (97) in which those complements are embedded are not exact paraphrases, however. The *that*-complement sentence allows a reading in which two separate babies are being discussed, while for sentence (92) there is only one baby who is an argument of both PERSUADE and GO. This difference in interpretation is more obvious when quantifiers

are involved: (104a) and (104b) are roughly synonymous, and neither is equivalent to (104c).

(104)

a. The girl persuaded every baby to go.
b. The girl persuaded every baby that he should go.
c. The girl persuaded every baby that every (other) baby should go.

Since semantic translation is defined on functional structure, f-structures must mark the difference between occurrences of similar subsidiary f-structures where semantic coreferentiality is implied, as in the *to*-complement, and occurrences where the similarity is only accidental.

The necessary f-structure distinction follows from a simple formal property of semantic forms that we now introduce. The semantic form representations that appear in schemata are treated as "meta" semantic forms, templates for an infinite number of distinct "actual" semantic forms. Just as an actual variable is substituted for a metavariable by the instantiation procedure, so a metaform is replaced by a unique actual form, identified by attaching an index to the predicate-argument specification. A given schema, say (105a), might be instantiated as (105b) at one node in the tree and (105c) at another:

(105)

a. $(\uparrow \text{PRED}) = \text{`BABY'}$
b. $(f_4 \text{ PRED}) = \text{`BABY'}_1$
c. $(f_6 \text{ PRED}) = \text{`BABY'}_2$

F-description statements and f-structures thus contain recognizably distinct instances of the semantic forms in the grammar and lexicon. Each indexed actual form enters into predicate-argument relations as indicated by the metaform, but the different instances are not considered identical for the purposes of semantic translation or functional uniqueness.

Returning to the two complements of *persuaded*, we observe that only one schema with 'BABY' is involved in the derivation of the *to*-complement, while two such schemata are instantiated for the *that*-complement. The indices of the two occurrences of 'BABY' are therefore the same in the indexed version of the *to*-complement's f-structure (106) but different in the f-structure for the *that*-complement (107). ((107) ignores such details as the tense and mood of the *that*-complement.)

(106)

$$\left[\begin{array}{ll}
\text{SUBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`GIRL'}_1\end{array}\right]\\
\text{TENSE} & \text{PAST}\\
\text{PRED} & \text{`PERSUADE}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ})\ (\uparrow \text{VCOMP})\rangle\text{'}_2\\
\text{OBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{THE}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`BABY'}_3\end{array}\right]\\
\text{VCOMP} & \left[\begin{array}{ll}\text{SUBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{THE}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`BABY'}_3\end{array}\right]\\ \text{INF} & +\\ \text{TO} & +\\ \text{PRED} & \text{`GO}\langle(\uparrow \text{SUBJ})\rangle\text{'}_4\end{array}\right]
\end{array}\right]$$

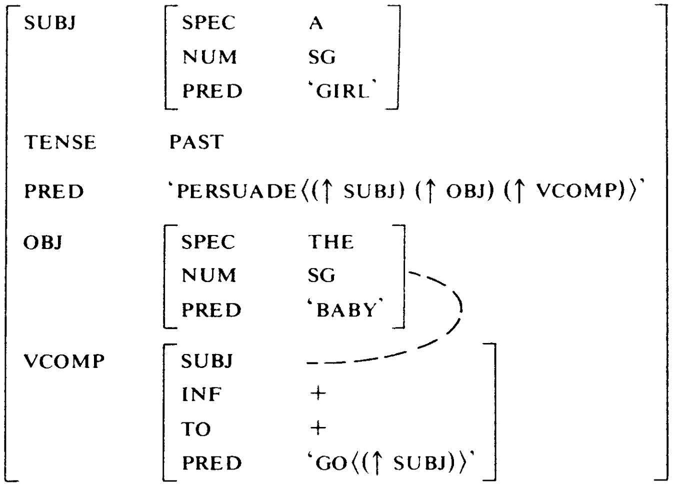
(107)

$$\left[\begin{array}{ll}
\text{SUBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{A}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`GIRL'}_1\end{array}\right]\\
\text{TENSE} & \text{PAST}\\
\text{PRED} & \text{`PERSUADE}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ})\ (\uparrow \text{VCOMP})\rangle\text{'}_2\\
\text{OBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{THE}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`BABY'}_3\end{array}\right]\\
\text{SCOMP} & \left[\begin{array}{ll}\text{SUBJ} & \left[\begin{array}{ll}\text{SPEC} & \text{THE}\\ \text{NUM} & \text{SG}\\ \text{PRED} & \text{`BABY'}_5\end{array}\right]\\ \text{PRED} & \text{`GO}\langle(\uparrow \text{SUBJ})\rangle\text{'}_4\end{array}\right]
\end{array}\right]$$

The semantic contrast between the two complement types is marked in these f-structures by the differing patterns of semantic form indexing.

It is technically correct to include indices with all semantic forms in f-descriptions and f-structures, but the nonidentity of two forms with dissimilar predicate-argument specifications is clear even without explicit indexing. We adopt the following convention to simplify our representations: two semantic form occurrences are assumed to be distinct unless they have the same predicate-argument specification and the same index. With this convention, only the indices on the BABY semantic forms are necessary in (106), and none of the indices are needed in (107). Control equations imply that entire substructures to which coindexed semantic forms belong will appear redundantly in several positions in an enclosing f-structure. This suggests a stronger abbreviatory convention which also highlights the cases of f-structure identity. The internal properties of a multiply-appearing subsidiary f-structure are displayed at only one place in an enclosing f-structure. The fact that it is also the value of other attributes is then indicated by drawing lines from the location of those other attributes to the fully expanded value:

(108)



This graphical connection makes it clear even without explicit indices on 'BABY' that the object f-structure serves in several functional roles.

While a semantic form instance occurring in several positions indicates semantic coreferentiality, different instances are seen as both semantically and functionally distinct. This means that any attempt to equate different instances will violate the Uniqueness Condition, even if they have the same predicate-argument specification. This is an important consequence of the semantic form instantiation procedure. For example, it rules out an analysis of string (109) in which both prepositional phrases are merged together as the BY OBJ, even though the PP f-structures agree in all other features:

(109)
*The baby was given a toy by the girl by the girl.

As another example, the distinctness of semantic form instances permits a natural description of English subject–auxiliary inversion. As shown in (110), the auxiliary can occur either before or after the subject, but it must appear in one and not both of those positions.

(110)
a. A girl is handing the baby a toy.
b. Is a girl handing the baby a toy?
c. *A girl the baby a toy.
d. *Is a girl is handing the baby a toy?

In transformational theories, facts of this sort are typically accounted for by a rule that moves a single base-generated item from one position to another. Since no transformational apparatus is included in LFG, we must allow the c-structure grammar to optionally generate the auxiliary in both positions, for example, by means of the following modified S and VP rules:

(111)

a. $$S \rightarrow \begin{pmatrix} \text{V} \\ (\downarrow \text{AUX}) =_c + \end{pmatrix} \quad \begin{matrix} \text{NP} \\ (\uparrow \text{SUBJ}) = \downarrow \\ (\downarrow \text{CASE}) = \text{NOM} \end{matrix} \quad \begin{matrix} \text{VP} \\ \uparrow = \downarrow \\ (\uparrow \text{TENSE}) \end{matrix}$$

b. $$\text{VP} \rightarrow$$
$$(\text{V}) \begin{pmatrix} \text{NP} \\ (\uparrow \text{OBJ}) = \downarrow \end{pmatrix} \begin{pmatrix} \text{NP} \\ (\uparrow \text{OBJ2}) = \downarrow \end{pmatrix} \quad \begin{matrix} \text{PP*} \\ \begin{Bmatrix} (\uparrow (\downarrow \text{PCASE})) = \downarrow \\ \downarrow \in (\uparrow \text{ADJUNCTS}) \end{Bmatrix} \end{matrix} \quad \begin{pmatrix} \text{VP}' \\ (\uparrow \text{VCOMP}) = \downarrow \end{pmatrix}$$

These rules provide c-structure derivations for all the strings in (110). However, (110c) is incoherent because there are no PREDs for the NP arguments, and it also fails to satisfy the TENSE existential constraint. The f-description for (110d) is inconsistent because the separately instantiated semantic forms for *is* are both assigned as its PRED. The AUX constraint in (111a) permits only verbs marked with the AUX feature to be fronted.

In section 4.5 we treated the auxiliary *is* as a main verb taking an embedded VP complement with a control schema identifying the matrix and embedded subjects (see (70)). *Is* is unlike *persuaded* and *promised* in that the f-structure serving two functional roles is not an argument of two predicates: SUBJ does not appear in the semantic form 'PROG⟨(↑ VCOMP)⟩'. The wider class of raising verbs differs from equi verbs in just this respect. Thus, the lexical entry for PERSUADE maps the *baby* f-structure in (108) into argument positions of both PERSUADE and GO. The OBJ of the raising verb *expected*, however, is an argument only of the complement's predicate, as stipulated in the lexical entry (112):

(112)

$$\begin{aligned} \text{expected: V, } & (\uparrow \text{TENSE}) = \text{PAST} \\ & (\uparrow \text{PRED}) = \text{`EXPECT}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{VCOMP})\rangle\text{'} \\ & (\uparrow \text{VCOMP TO}) =_c + \\ & (\uparrow \text{VCOMP SUBJ}) = (\uparrow \text{OBJ}) \end{aligned}$$

Except for the semantic form change, the f-structure for sentence (113a) is identical to (108). This minor change is sufficient to account for the well-known differences in the behavior of these two classes of verbs, as illustrated by (113b) and (113c) (see chapter 1 for a fuller discussion).

(113)
a. The girl expected the baby to go.
b. The girl expected there to be an earthquake.
c. *The girl persuaded there to be an earthquake.

The difference between the raising and equi semantic forms shows that the set of grammatical relations in an f-structure cannot be identified with argument positions in a semantic translation. This is evidence for our early claim that the functional level is also distinct from the semantic level of representation. A stronger justification for this distinction comes from considerations of quantifier scope ambiguities. The sentence (114a) has a single f-structure, yet it has two semantic translations or interpretations, corresponding to the readings (114b) and (114c):

(114)
a. Every man voted in an election.
b. 'There was an election such that every man voted in it.'
c. 'For every man there was an election such that he voted in it.'

The *election* quantifier has narrow scope in (114b) and wide scope in (114c). This ambiguity is not represented at the level of syntactic functions since no syntactic generalizations depend on it. Instead, the alternative readings are generated by the procedure that produces semantic translations or interpretations for f-structures. (This line of argumentation was suggested by P. K. Halvorsen (personal communication). Halvorsen 1980 and forthcoming gives a detailed description of a translation procedure with multiple outputs.)

The distinctions among c-structure, f-structure, and semantic structure are supported by another scope-related phenomenon. Sentence (115a) also has two readings, as indicated in (115b) and (115c):

(115)
a. Everybody has wisely selected their successors.
b. 'Wisely, everybody has selected their successors (i.e., it is wise of everybody to have selected their successors).'
c. 'Everybody selected their successors in a wise manner.'

The adverb has sentence scope in (115b) and so-called VP scope in (115c). The single f-structure for this sentence not only fails to represent the ambiguity but also fails even to preserve a VP unit to which the narrow scope might be attached. The f-structure is flattened to facilitate the statement of certain syntactic cooccurrence restrictions, to simplify the Completeness and Coherence Conditions, as mentioned above, and also to permit simple specifications of control relations. Independent motivation for our proposal that the scope of semantic operators is not tied to a VP c-structure node or an f-structure corresponding to it comes from Modern Irish, a VSO language that nonetheless exhibits this kind of ambiguity (McCloskey 1979).

We have shown that functional structure in LFG is an autonomous level of linguistic description. Functional structure contains a mixture of syntactically and semantically motivated information, but it is distinct from both constituent structure and semantic representation. Of course, we have not demonstrated the necessity of such an intermediate level for mapping between surface sequences and predicate-argument relations. Indeed, Gazdar to appear b argues that a much more direct mapping is possible. In Gazdar's approach, the semantic connection between a functional controller and controllee, for example, is established by semantic translation rules defined directly on c-structure configurations. The semantic representation for the embedded complement includes a logical variable that is bound to the controller in the semantic representation of the matrix. It seems, however, that there are language-particular and universal generalizations that have no natural expression without an f-structure-like intermediate level. For example, in addition to semantic connections, functional control linkages seem to transmit purely syntactic elements—expletives like *it* and *there*, syntactic case-marking features (chapter 7 of this volume and Andrews to appear), and semantically empty idiom chunks. Without an f-structure level, either a separate feature propagation mechanism must be introduced to handle this kind of dependency in the c-structure, or otherwise unmotivated semantic entities or types must be introduced so that semantic filtering mechanisms can be applied to the syntactic elements. As another example, it is argued in chapter 9 of this volume that a natural account of sluicing constructions requires the mixture of information found in f-structures. And finally, it is observed in chapters 1, 5, and 8 of this volume and in Mohanan to appear, that universal characterizations of lexical rules and rules of anaphora are stated more naturally in terms of grammatical functions than in terms of phrase structure configurations or properties of semantic representations. Further investigation should provide even stronger justification for functional structure as an essential and independent level of linguistic description.

### 4.7 Long-Distance Dependencies

We now turn to the formal mechanisms for characterizing the long-distance grammatical dependencies such as those that arise in English questions and relatives. As is well known, in these constructions an element at the front of a clause is understood as filling a particular grammatical role within the clause. Exactly which grammatical function it serves is determined primarily by the arrangement of c-structure nodes inside the clause. The *who* before the indirect question clause is understood as the subject of the question in (116a) but as the object in (116b):

(116)
a. The girl wondered who ____ saw the baby.
b. The girl wondered who the baby saw ____.
c. *The girl wondered who ____ saw ____.
d. *The girl wondered who the baby saw the toy.

In both cases, *who* is assigned the clause-internal function appropriate to the c-structure position marked by the blank, a position where an expected element is missing. Examples (116c,d) indicate that there must be one and only one missing element. Sentence (117), in which the *who* is understood as the object of a clause embedded inside the question, shows the long-distance nature of this kind of dependency:

(117)
The girl wondered who John believed that Mary claimed that the baby saw ___.

Sentence (118), however, demonstrates the well-known fact that the regions of the c-structure that such dependencies may cover are limited in some way, although not simply by distance:

(118)
*The girl wondered who John believed that Mary asked who ___ saw ___.

The dependencies illustrated in these sentences are examples of what we call *constituent control*. As with functional control, constituent control establishes a syntactic identity between elements that would otherwise be distinct. (The term *syntactic binding* is sometimes used as a synonym for *constituent control*.) In the case of functional control, the linkage is between the entities filling particular functional roles and, as described in section 4.6, is determined by lexical schemata that are very restricted substantively. Functional control schemata identify particular functions (such as SUBJ or OBJ) at one f-structure level with the SUBJ of a particular complement. Linkages over apparently longer distances, as in (119), are decomposed into several strictly local identifications, each of which links a higher function to the SUBJ one level down.

(119)
John persuaded the girl to be convinced to go.

The f-description for this example contains statements that equate the OBJ of *persuaded* with the SUBJ of *be*, the SUBJ of *be* with the SUBJ of *convinced*, and finally the SUBJ of *convinced* with the SUBJ of *go*. The fact that *girl* is understood as the subject of *go* then follows from the transitivity of the equality relation. However, it is characteristic of functional control that *girl* also bears grammatical relations to all the intermediate verbs, and that the intermediate verbs necessarily carry the required control schemata. A long-distance functional linkage can be made unacceptable by an intermediate lexical change that has no c-structure consequences:

(120)
a. There was expected to be an earthquake.
b. *There was persuaded to be an earthquake.

The f-structure becomes semantically incomplete when the equi verb *persuaded* is substituted for the intervening raising verb.

Constituent control differs from functional control in that constituent structure configurations, not functional relations, are the primary conditioning factors. As noted in (112), at one end of the linkage (called the *constituent controllee*), the clause-internal function may be determined by the position of a c-structure gap. The relative clause in (121) demonstrates that the c-structure environment alone can also define the other end of the linkage (called the *constituent controller*):

(121)
The toy the girl handed ___ to the baby was big.

This sentence has no special words to signal that *toy* must enter into a control relationship. Finally, the linked entity bears no grammatical relation to any of the predicates that the constituent dependency covers (e.g., *believed* and *claimed* in (117)), and there are no functional requirements on the material that may intervene between the controller and the controllee. Instead, the restrictions on possible linkages involve the configuration of nodes on the controller–controllee c-structure path: for example, the interrogative complement of *asked* on the controller–controllee path in (118) is the source of that string's ungrammaticality.

Decomposing these long-distance constituent dependencies into chains of functional identifications would require introducing otherwise unmotivated functions at intermediate f-structure levels. Such a decomposition therefore cannot be justified. A strategy for avoiding spurious functions is to specify these linkages by sets of alternative direct functional identifications. One alternative would link the *who* to the SUBJ of the clause for (116a), and a second alternative would link to the OBJ for (116b). Question clauses with one embedded sentential complement would require alternatives for the SCOMP SUBJ and SCOMP OBJ; the two embeddings in (117) would require SCOMP SCOMP OBJ; and so on. This strategy has an obvious difficulty: without a bound on the functional distance over which this kind of dependency can operate, the necessary alternative identifications cannot be finitely specified.[20] The functional apparatus of our theory thus does not permit an adequate account of these phenomena.

If a single constituent contains no more than one controllee, it is possible to encode enough information in the c-structure categories to ensure a correspondence between controllers and controllees, as suggested by Gazdar to appear b. This encoding obviously captures the fact that these dependencies are sensitive to constituent configurations. Gazdar also shows that appropriate semantic representations can be defined by translations associated with the phrase structure rules. Maling and Zaenen 1980 point out that this approach becomes considerably less attractive if a single constituent can contain more than one controllee, as in the familiar interaction of *tough*-movement and questions in English:

(122)
I wonder which violin the sonatas are easy to play ___ on ___.

Furthermore, no encoding into a finite number of categories is possible for languages such as Swedish and Norwegian, for which, according to Maling and Zaenen to appear and Engdahl 1980a,b, no natural limit can be set on the number of controllees in a single constituent.

Our problem, then, is to provide a formal mechanism for representing long-distance constituent dependencies that does not require unmotivated grammatical functions or features, allows for an unbounded number of controllees in a single constituent, and permits a succinct statement of the generalizations that govern grammatical phenomena of this sort. The necessary descriptive apparatus is found in the formal interpretation of *bounded domination metavariables*.

The bounded domination metavariables $\Uparrow$ and $\Downarrow$ are similar to the immediate domination variables $\uparrow$ and $\downarrow$ in that they appear in grammatical and lexical schemata but are instantiated with actual variables when the f-description is formed. The instantiation procedure for both kinds of variables has the effect of substituting the same actual variable for matched metavariables attached to different nodes in the c-structure. The difference is that for a matched $\uparrow$–$\downarrow$ pair, the schemata must be attached to nodes in a relationship of immediate domination, while matching $\Downarrow$ and $\Uparrow$ may be attached to nodes separated in the tree by a longer path. These are called "bounded domination metavariables" because that path is limited by the occurrence of certain "bounding" nodes. The $\Downarrow$ metavariable is attached to a node at the upper end of the path and represents the controller of a constituent control relationship. (Technically, the terms *controller* and *controllee* refer to the bounded domination metavariables and not to the nodes that they are attached to. In this respect, we depart from the way these terms have been used in other theoretical frameworks.) The matching $\Uparrow$ is lower in the tree and represents the controllee of the relationship. The instantiation procedure for these variables establishes the long-distance identification of the controller and controllee directly, without reliance on transitive chains of intervening equations.

We illustrate the general properties of our mechanism with a simple example, suppressing for the moment a number of formal and linguistic details. Consider the indirect question sentence (116b), repeated here for convenience:

(116)
b. The girl wondered who the baby saw ___.

We assume that the predicate for *wondered* takes an interrogative complement argument, as indicated in the lexical entry (123):[21]

(123)
wondered: V, ($\uparrow$ TENSE) = PAST
($\uparrow$ PRED) = 'WONDER⟨($\uparrow$ SUBJ) ($\uparrow$ SCOMP)⟩'

According to the rules in (124), SCOMPs are based on constituents in the category S′, and S′ expands as an NP followed by an S:

(124)

$$\text{a.}\quad \text{VP} \rightarrow \begin{array}{cc} \text{V} & \text{S}' \\ & (\uparrow \text{SCOMP}){=}\downarrow \end{array}$$

$$\text{b.}\quad \text{S}' \rightarrow \begin{array}{cc} \text{NP} & \text{S} \\ (\uparrow \text{Q-FOCUS}){=}\downarrow & \uparrow{=}\downarrow \\ \downarrow{=}\Downarrow & \end{array}$$

The schemata in (124b) mark the initial NP as the question's focus (Q-FOCUS) and also identify it with $\Downarrow$, the controller of a gap in the following S. The initial NP for our example is realized as the interrogative pronoun *who*, which has the following lexical entry:

(125)
who: N, ($\uparrow$ PRED) = 'WHO'

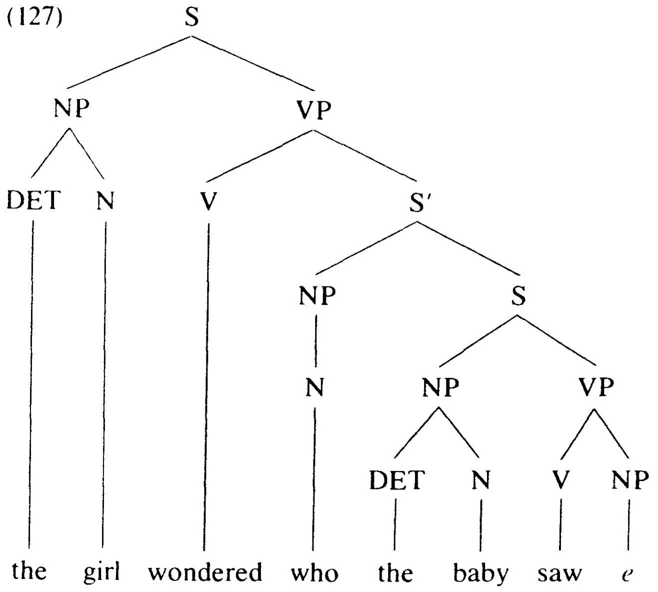
The final rule for this example associates the controllee metavariable $\Uparrow$ with a gap position inside the clause. As shown in (126), we allow c-structure rules to expand a nonterminal category as the empty string, symbolized by *e*. This gives a formal representation for the intuition that an element of that category is missing.

(126)

$$NP \rightarrow \underset{\uparrow=\Uparrow}{e}$$

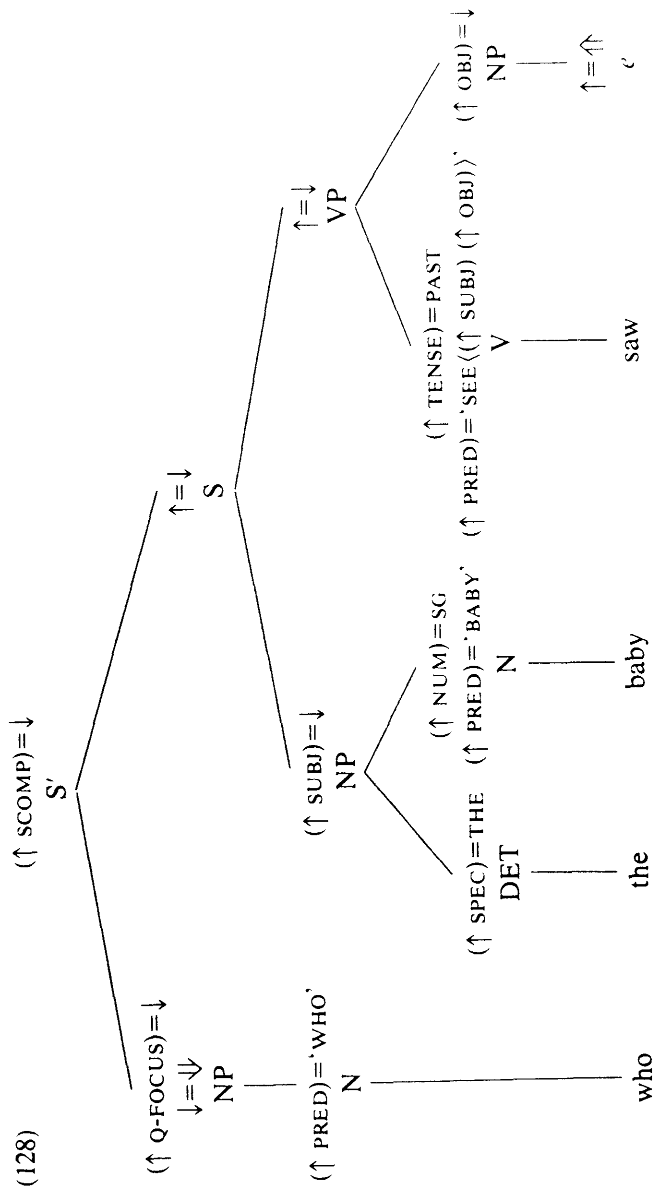
The schema on the empty expansion introduces the controllee metavariable.[22] This NP alternative must be utilized for the object of *saw* so that (116b) is assigned the c-structure (127):

(127)



The instantiation procedure for metavariables still has an attachment phase, a variable introduction phase, and a substitution phase, just as it was presented in section 4.3. Schemata are attached to appropriate c-structure nodes in the first phase without regard to the kinds of metavariables they contain. The attachments for nodes in the embedded S′ subtree are shown in (128). In the second phase, distinct actual variables are introduced for the root node and for every node where a schema contains a ↓ metavariable. This provides the ↓-variables for the nodes, as before. However, an additional variable is introduced for each node with a schema containing the controller metavariable ⇓, providing a ⇓-variable for that node. For this simple example, only the *who* NP node has a controller and receives the extra variable assign-

(128)



ment. The annotations $\downarrow{:}f_5$ and $\Downarrow{:}f_6$ on that node in (129) record the association between metavariables and actual variables. For immediate domination metavariables, the instantiation is completed by substituting a node's $\downarrow$-variable for all the $\downarrow$'s at that node and for all corresponding $\uparrow$'s, those in schemata attached to its daughter nodes. The treatment of bounded domination metavariables is similar in that the $\Downarrow$-variable of a node replaces all the $\Downarrow$'s at that node and all corresponding $\Uparrow$'s. The essential difference is that the nodes to which corresponding $\Uparrow$'s are attached may be further away in the c-structure.

The $\Uparrow$ corresponding to the $\Downarrow$ on the *who* NP in (129) is attached to the empty object of *saw*. The substitution phase of instantiation thus adds the following statements to the f-description:

(130)

a. $(f_4\ \text{Q-FOCUS}) = f_5$

b. $f_5 = f_6$

c. $(f_5\ \text{PRED}) = \text{'WHO'}$

d. $(f_9\ \text{OBJ}) = f_{10}$

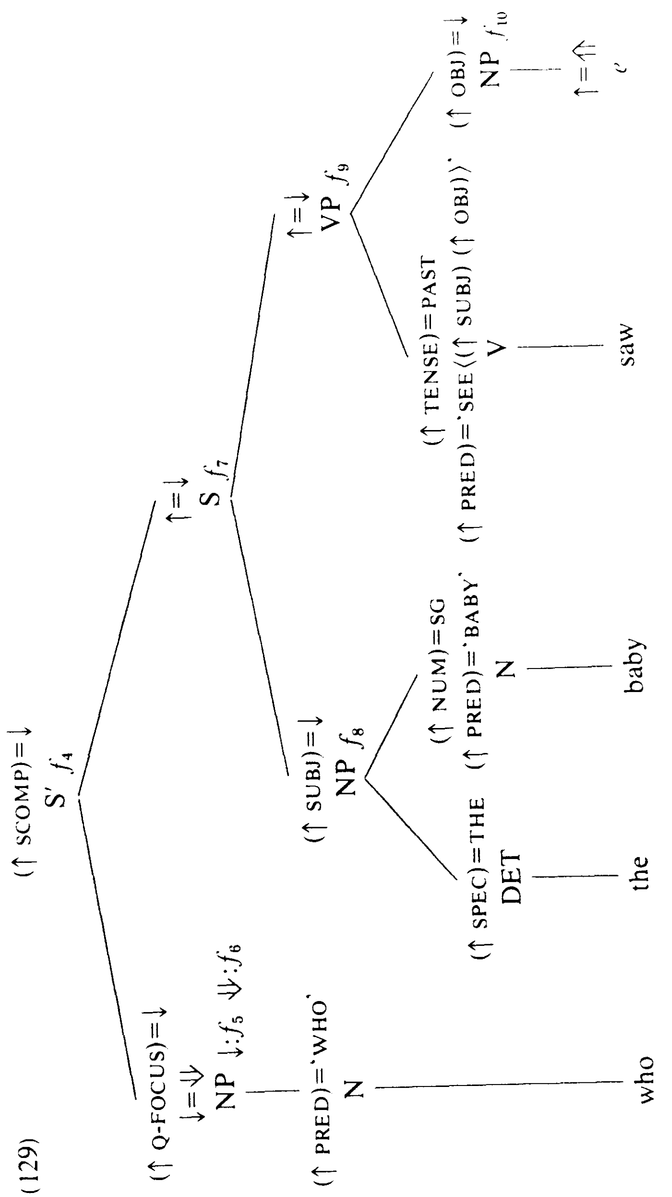
e. $f_{10} = f_6$

Equation (130b) comes from the *who* NP node and (130e) comes from the empty NP expansion. Both equations contain the $\Downarrow$-variable $f_6$ and thereby establish the crucial linkage: the semantic form 'WHO' serves as the PRED in the object f-structure for *saw* and accounts for the fact that *who* is understood as the second argument of SEE. This is apparent in f-structure (131), the solution to sentence (116b)'s f-description:

(131)

$$\left[\begin{array}{ll}
\text{SUBJ} & \left[\begin{array}{ll} \text{SPEC} & \text{THE} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{'GIRL'} \end{array}\right] \\
\text{TENSE} & \text{PAST} \\
\text{PRED} & \text{'WONDER}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{SCOMP})\rangle\text{'} \\
\text{SCOMP} & \left[\begin{array}{ll}
\text{Q-FOCUS} & \left[\begin{array}{ll} \text{PRED} & \text{'WHO'} \end{array}\right] \\
\text{SUBJ} & \left[\begin{array}{ll} \text{SPEC} & \text{THE} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{'BABY'} \end{array}\right] \\
\text{TENSE} & \text{PAST} \\
\text{PRED} & \text{'SEE}\langle(\uparrow \text{SUBJ})\ (\uparrow \text{OBJ})\rangle\text{'} \\
\text{OBJ} & \text{------------}
\end{array}\right]
\end{array}\right]$$

(129)

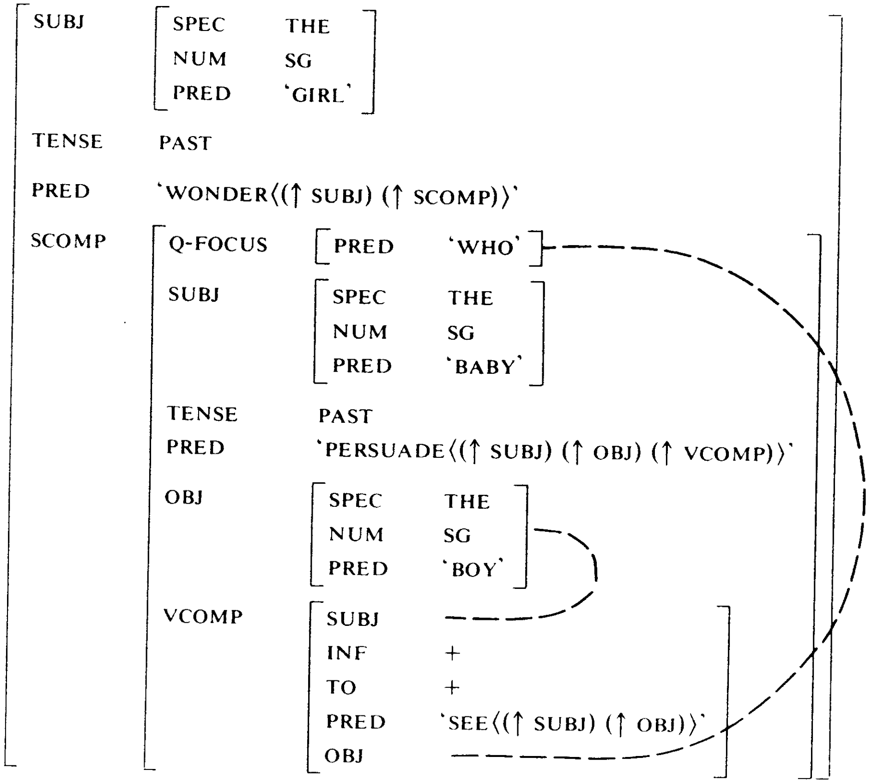


Thus, constituent control dependencies are handled in LFG by extending the instantiation procedure for mapping schemata on c-structure nodes into f-description statements. Because we do not rely on intermediate functional identifications, the statements in (130) are sufficient to establish the same connection over longer c-structure distances, for example, over the intervening *to*-complement in (132):

(132)
The girl wondered who the baby persuaded the boy to see ___.

Except for possibly a different choice of actual variables, the instantiation procedure would again produce the statements (130), correctly representing the constituent control relation. The f-structure for this sentence has both a functional control linkage and a constituent control linkage:

(133)



Note that there are no extraneous attributes or values to carry the constituent control linkage through the PERSUADE f-structure.

The instantiation procedure as described substitutes the same actual variable for a $\Downarrow$ and any "corresponding" $\Uparrow$'s. Beneath this vague notion of correspondence lies some additional notation and a rich set of definitions and restrictions that we now make precise. We observe first that corresponding $\Downarrow$'s and $\Uparrow$'s must meet certain category requirements. As examples (134a,b) indicate, the verb *grow* meaning 'become' may be followed by an adjective phrase but not an NP, while the verb *reach* meaning 'extend to' has just the opposite distribution. Example (134c) shows that a controller may be associated with an AP at the beginning of an indirect question, but its corresponding controllee must then be in an adjectival position. Example (134d) demonstrates that metavariables associated with NPs must also be compatible:

(134)
a. She'll grow that tall/*height.
b. She'll reach that *tall/height.
c. *The girl wondered how tall she would grow/*reach ___.
d. *The girl wondered what height she would *grow/reach ___.

We therefore allow bounded domination metavariables to carry specifications of c-structure categorial features. These specifications are written as subscripts on the metavariables, and we require that corresponding controllers and controllees have compatible subscripts. Thus, a $\Downarrow_{NP}$ may correspond to a $\Uparrow_{NP}$ but not to a $\Uparrow_{AP}$. The contrast in (134d) then follows from adding the subscript NP to the metavariables in our previous rules:

(135)

a. $$\begin{array}{lcc} \text{S}' \rightarrow & \text{NP} & \text{S} \\ & (\uparrow \text{Q-FOCUS})=\downarrow & \uparrow=\downarrow \\ & \downarrow=\Downarrow_{\text{NP}} & \end{array}$$

b. $$\begin{array}{lc} \text{NP} \rightarrow & e \\ & \uparrow=\Uparrow_{\text{NP}} \end{array}$$

The rules for handling adjectival and prepositional dependencies have analogous categorial markings, and cross-categorial correspondences are thereby excluded.

For these examples, the categorial subscripts are redundant with the categories of the nodes that the metavariables are associated with, but this is not always the case. In (136a) the metavariable associated with the topicalized S′ is matched with a controllee on an *e* in a c-structure NP position, a prepositional object. (136b) rules out the possibility that the S′ is dominated by an NP. The contrast between (136c) and (136d) shows that a topicalized S′ cannot control an S′ c-structure position.

(136)
a. That he might be wrong he didn't think of ___.
b. *He didn't think of that he might be wrong.
c. He didn't think that he might be wrong.
d. *That he might be wrong he didn't think ___.

This pattern follows directly from associating a $\Downarrow_{NP}$ metavariable with the fronted S′ node.

Another obvious property of acceptable correspondences is that certain tree relations must hold between the nodes to which corresponding controller and controllee metavariables are attached. The *e* corresponding to the *who* controller in (129) must be dominated by the adjacent S node. It cannot be located earlier or later in the main clause, nor inside a more complicated NP in the *who* position. To put it in more technical terms, we say that the S node in (129) is the root of a *control domain* for the *who* $\Downarrow_{NP}$. For a controller attached to a given node in the c-structure, a control domain consists of the nodes in a subtree that a corresponding controllee may be attached to. Our notion of corresponding metavariables thus turns on a rigorous characterization of what nodes can be roots of control domains and what nodes dominated by the root are contained in the domain.

A controller metavariable carries still another specification that determines what node may be its domain root. A closer examination of the indirect question construction shows why this is needed. Rule (135a) suggests that any NP may appear at the front of an indirect question, but this is of course not the case. The fronted phrase is restricted to contain an interrogative word of some sort. That word need not be at the top level of the NP as in (116b), but may rather be deeply embedded within it:

(137)
The girl wondered whose playmate's nurse the baby saw ___.

This sentence would be generated by the alternative NP rule (138), which allows for possessors with genitive case in prenominal position. (We assume that morphological rules correlate the genitive case marking with the *'s* suffix, and that *whose* is morphologically composed of *who* + *'s*.)

(138)

$$\begin{array}{lll} \text{NP} \rightarrow & \text{NP} & \text{N} \\ & (\downarrow \text{CASE}) =_c \text{GEN} & \\ & (\uparrow \text{POSS}) = \downarrow & \end{array}$$

A very natural way of guaranteeing the presence of a question word in the appropriate contexts is to specify a constituent control relation between the fronted NP of an indirect question and the interrogative embedded underneath it. This is possible because constituent control in our theory may affect not only null elements but also a designated set of lexical items which includes interrogative pronouns, determiners, and adverbs.

Even though interrogative elements differ in their major categorial features, we assume that they are distinguished from other lexical items by the appearance of a morphosyntactic feature [+wh] in their categorial feature matrices, and we use this feature as the metavariable subscript for the interrogative constituent control dependency. However, it is not sufficient to revise our previous S′ rule simply by adding a [+wh] controller metavariable to the fronted NP:

(139)

$$\begin{array}{lll} \text{S}' \rightarrow & \text{NP} & \text{S} \\ & (\uparrow \text{Q}) = \Downarrow_{[+\text{wh}]} & \uparrow = \downarrow \\ & (\uparrow \text{FOCUS}) = \downarrow & \\ & \downarrow = \Downarrow_{\text{NP}} & \end{array}$$

When the schemata from this rule are attached to the nodes in sentence (137)'s c-structure, two different controllers, $\Downarrow_{NP}$ and $\Downarrow_{[+wh]}$, are associated with the fronted NP node. While we still intend the S to be the domain root for the $\Downarrow_{NP}$, we intend the root for $\Downarrow_{[+wh]}$ to be the fronted NP itself. In order to represent this distinction, we must explicitly mark the individual controllers with category symbols that determine their respective domain roots. The superscript S in the controller $\Downarrow^{S}_{NP}$ indicates that the corresponding $\Uparrow_{NP}$ must be found in an S-rooted control domain, while the [+wh] controllee for $\Downarrow^{NP}_{[+wh]}$ must be found beneath an NP node. Moreover, the domain roots must be either the nodes to which the controllers are attached or sisters of those nodes, as indicated in the following definition:

(140)
*Root Node of a Constituent Control Domain*

Suppose $\Downarrow^{r}$ is a controller metavariable attached to a node $N$. Then a node $R$ is the root node of a control domain for $\Downarrow^{r}$ if and only if
a. $R$ is a daughter of $N$'s mother, and
b. $R$ is labeled with category $r$.

Introducing root-category superscripts into the S′ rule, we have:

(141)

$$\begin{array}{lll} S' \rightarrow & \text{NP} & \text{S} \\ & (\uparrow \text{Q})=\Downarrow^{\text{NP}}_{[+\text{wh}]} & \uparrow=\downarrow \\ & (\uparrow \text{FOCUS})=\downarrow & \\ & \downarrow=\Downarrow^{\text{S}}_{\text{NP}} & \end{array}$$

The [+wh] controllee for the interrogative linkage is associated with a lexically realized N node, not with an empty string expansion, and the schema containing the controllee metavariable does not come from the grammar but rather from the lexical entry for *who:*

(142)
who: N, $(\uparrow \text{PRED}) = \text{`WHO'}$
$\uparrow = \Uparrow_{[+\text{wh}]}$

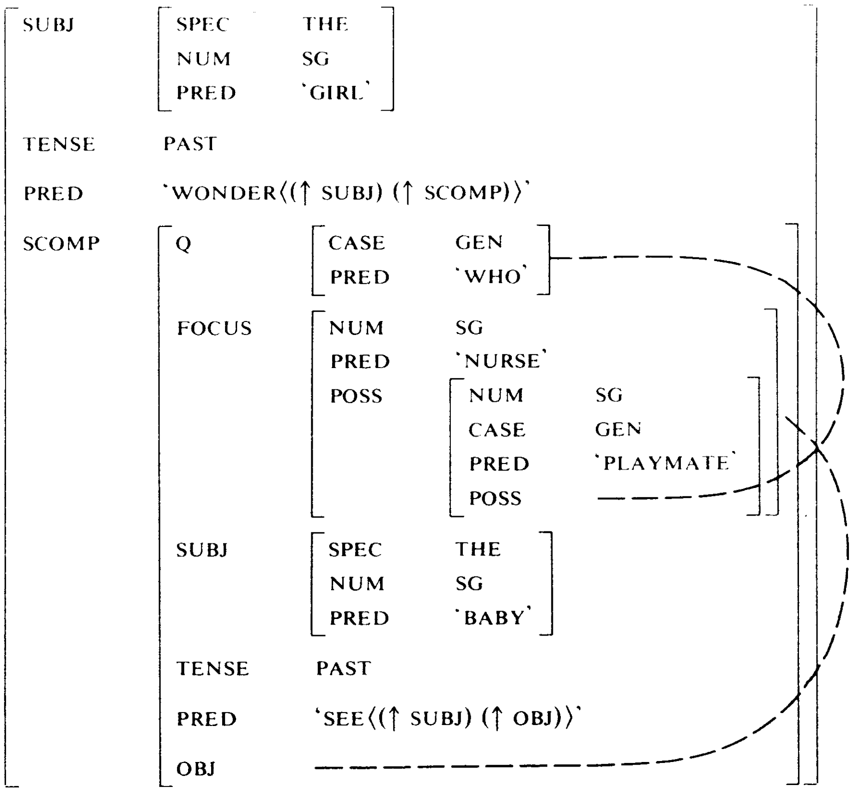
The lexical entry and our revised question rule yield the f-structure (143) for sentence (137).[23]

The root-node category specification provides one part of the characterization of what a control domain can be. To complete this characterization, we must define which nodes dominated by the domain root are contained in the domain. The *wh*-island in example (144) demonstrates that at least some nodes in the domain root's subtree do not belong to the domain:

(144)
*The girl wondered what the nurse asked who ___ saw ___.

Without some limitation on the extent of a domain, $\Uparrow_{\text{NP}}$'s at the gaps would be interpretable as the controllees for *who* and *what*, respectively. Limitations on what nodes may belong to a given control domain come from the fact that nodes in certain c-structure configurations are classified as *bounding nodes*. The path from a node in a domain to the domain root is then restricted as in (145).

(143)



(145)
*Bounding Convention*

A node $M$ belongs to a control domain with root node $R$ if and only if $R$ dominates $M$ and there are no bounding nodes on the path from $M$ up to but not including $R$.

The domain root thus carries a substantial theoretical burden as a c-structure intermediary between the nodes to which a controller metavariable and its corresponding controllees are attached. The categorial superscript on the controller metavariable is a direct and definite selector of its domain roots. However, the path from a root to a corresponding controllee's node, while restricted by the Bounding Convention, is not uniquely determined by the grammar.

It remains to extend our notion of grammaticality to take bounded domination metavariables explicitly into account. Intuitively, we require all controllers to have corresponding controllees and all controllees to have corresponding controllers, so that there are no uninstantiated metavariables in the f-description. We add the following to our previous list of grammaticality conditions:

(146)
*Grammaticality Condition*
A string is grammatical only if its f-description is *properly instantiated.*

The controller/controllee correspondence is one consequence of the formal definition of *proper instantiation:*

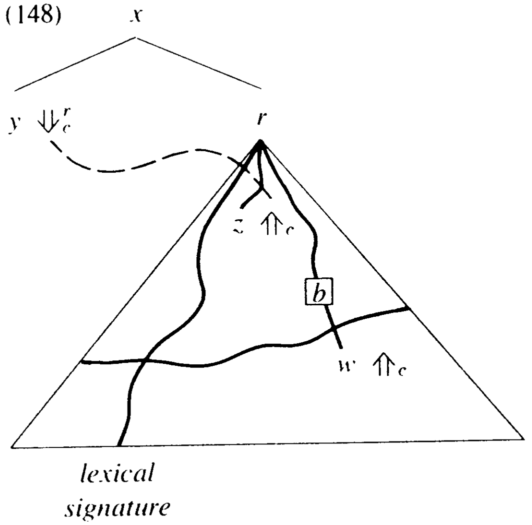
(147)
*Definition of Proper Instantiation*
The f-description from a c-structure with attached schemata is properly instantiated if and only if:
a. no node is a domain root for more than one controller,
b. every controller metavariable has at least one control domain,
c. every controller metavariable corresponds to one and only one controllee in each of its control domains,
d. every controllee metavariable corresponds to one and only one controller,
e. all metavariable correspondences are *nearly nested,* and
f. every domain root has a *lexical signature.*

For a properly instantiated f-description, there is a one-to-one mapping between controllees and domain roots, and each domain root is associated with one and only one controller. This establishes the necessary correspondence between metavariables. The definition of *nearly nested correspondences* and the consequences of the restriction (147e) are presented at the end of this section, where we discuss the possibility of a single constituent containing several controllees.

The lexical signature clause is motivated primarily by formal considerations. It establishes a connection between controlled *e*'s and actual lexical items that plays an important role in the recursiveness proof presented in section 4.8. For each domain root there must be a distinct word in the terminal string. This word is called the *lexical signature* of the domain root. The domain root must dominate its lexical signature. The effect of (147f) is that each domain root, and thus each control domain, must be reflected in the string in some unique way.[24] One possible interpretation of this formal condition is that a control domain must have a lexically realized "head." The head can be defined in terms of the X′ category system. It can also be defined purely in functional terms: a lexical head is the lexical item that contributes the PRED semantic form to a constituent's ↓ f-structure.

According to (147), corresponding metavariables of a grammatical sentence must be in a c-structure configuration as outlined in (148):

(148)



In this c-structure and in the illustrations below, bounding nodes are enclosed in boxes. The dashed line passes by the domain root to connect the corresponding controller and controllee. The lower $\Uparrow_c$ in (148) cannot correspond to the controller because the bounding node *b* lies on the path to the root *r*.

Bounding nodes define "islands" of the c-structure that constituent control dependencies may not penetrate. They serve the same descriptive purpose as Ross's 1967 constraints on transformational variables and Chomsky's 1977b notion of cyclic or bounding categories. Those theories, however, have descriptive inadequacies. Ross hypothesized that constraints such as the Complex NP Constraint apply to all human languages, but this has proved not to be the case. All Scandinavian languages, for example, permit long-distance dependencies to cross the boundaries of indirect questions, and all except for Icelandic permit them to cross the boundaries of relative clauses as well (for illustrations, see Erteschik 1973, Allwood 1976, Engdahl 1980a,b, Maling and

Zaenen to appear). Moreover, although dependencies into English relative clauses (149a) are unacceptable, Ross himself noted that extractions from phrases within the lexically filled NPs in examples like (149b,c) are possible even in English:

(149)

a. *I wonder who the man that ____ talked to ____ saw Mary.
b. I wonder who John saw a picture of ____.
c. Who was it that John denied the claim that he dated ____?

The restrictions on constituent control into English sentential complements and relative clauses seem to be governed by different generalizations; Godard forthcoming convincingly argues that a similar pattern holds for complements and relatives in French. In Chomsky's theory, the Subjacency Convention provides a general limitation on syntactic rules. The domains of rule application are thereby restricted by the occurrence of nodes in specified categories. Chomsky shows that many of the properties of English dependencies follow from the assumption that S′ and NP (and possibly S) are bounding categories. One reasonable extension to Chomsky's theory defines bounding categories on a language-by-language basis: stipulating a smaller (or perhaps empty) set of bounding categories in the grammar of Swedish might give an account of the freer dependencies exhibited by that language. However, the English sentences (149b,c) have no natural description in Chomsky's system if *all* NPs in English are bounding nodes.[25]

Bounding node specifications in lexical-functional grammar acknowledge the fact that restrictions on long-distance dependencies may vary between languages and between different nodes of the same category in particular languages. This flexibility does not diminish the explanatory potential of our formal system. We expect that a substantive theory of human language based on our formalism will stipulate a small, principled set of c-structure configurations in which bounding nodes may appear. The grammars of particular languages must draw from this universal inventory of possible bounding nodes to identify the bounding categories in individual c-structure rules (see Zaenen 1980 for some partial proposals). Further work will of course be needed to formulate and justify a universal bounding node theory. Our goal at present is only to illustrate the notation and formal properties of our constituent control mechanisms. A simple notational device is used to indicate that constituent control is blocked by nodes in particular c-structure configurations: enclosing a category on the righthand side of a c-structure rule in a box specifies that the nodes derived by that rule element are bounding nodes.

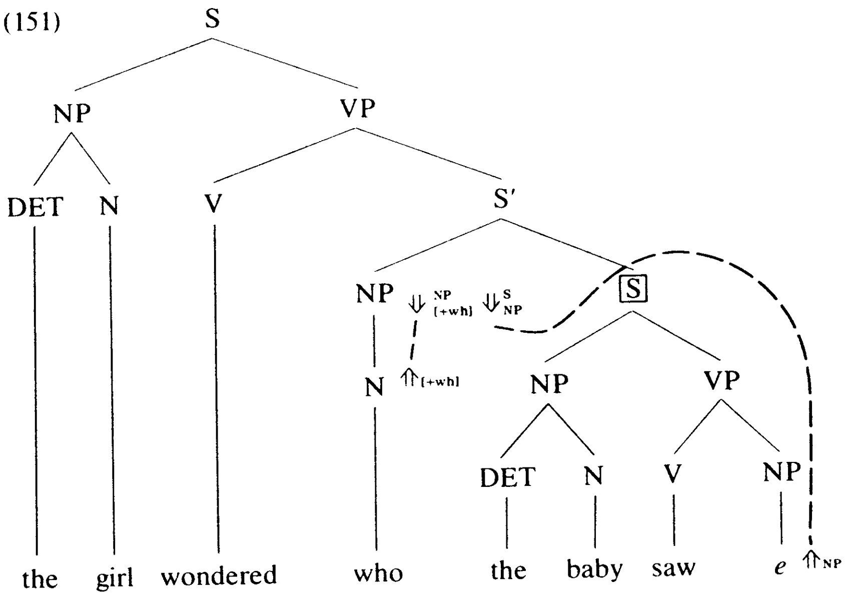
We incorporate this notation into our treatment of indirect questions for the variety of English in which they form islands. The S in these constructions is a bounding node, as shown in the revised rule:

(150)

$$S' \rightarrow \begin{array}{cc} NP & \boxed{S} \\ (\uparrow Q)=\Downarrow^{NP}_{[+wh]} & \uparrow=\downarrow \\ (\uparrow \text{FOCUS})=\downarrow & \\ \downarrow=\Downarrow^{S}_{NP} & \end{array}$$

Notice first that the bounding node introduced by this rule does not block the simple indirect question sentence (116b). As shown in (151), this is because the S is the root node of the controller's control domain. Therefore, in accordance with the Bounding Convention (145), it does not interfere with the metavariable correspondence.

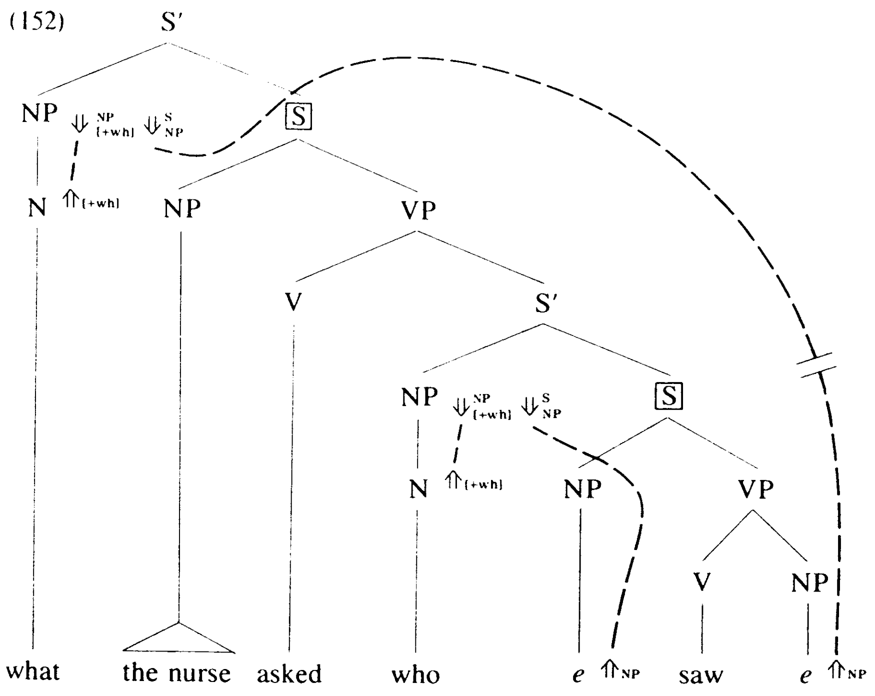
(151)



The dashed line in this illustration runs between the corresponding metavariables, not between the nodes they are attached to. The connected metavariables will be instantiated with the same actual variable.

The situation is different for the more complicated string (144). Neither of the gaps inside the *asked* question belongs to the control domain whose root node is the sister of *what*. This is because the *who* domain root is a bounding node on the path from each of the controllees to the root for the *what* $\Downarrow^{S}_{NP}$:

(152)



Conditions (147c,d) are not satisfied, and the string is marked ungrammatical.

Our box notation also permits an account of the apparent difference in NP bounding properties illustrated in (149). The S' in the relative clause expansion rule (153) is boxed, thus introducing a bounding node that separates both of the gaps in example (149a) from the *who* controller:

(153)
$NP \rightarrow NP\ \boxed{S'}$

A proper instantiation for this example is therefore impossible. Constituent control into the other NP constructions in (149) is acceptable because they are derived by alternative rules which do not generate bounding nodes. This distribution of bounding nodes has a further consequence. Together with our hypothesis that the interrogative word inside a fronted NP is subject to constituent control, it explains certain restrictions on the location of the interrogative. Sentence (154a) shows that a fronted NP may contain a relative clause, but example (154b) demonstrates that the interrogative pronoun may not appear *inside* the relative. This is just what we would predict, since the relative clause bounding node that separates the NP metavariables in (154c) also blocks the [+wh] correspondence in (154b):

(154)
a. The girl whose pictures of the man that called Mary I saw talked to John.
b. *The girl the pictures of the man that called whom I saw talked to John.
c. *The girl who I saw pictures of the man that called talked to John.

Though similar in these examples, there are English constructions in which NP and [+wh] metavariables do not have the same privileges of occurrence. We see in (155) and (156) that a controlled interrogative may, but a controlled *e* may not, be located in the possessive modifier of an NP:

(155)
The girl wondered whose nurse the baby saw ____.

(156)
*The girl wondered who the baby saw ____'s nurse.

The ungrammaticality of (156) follows from making the prenominal genitive NP be a bounding node, as in the revised NP rule (157):

(157)

$$NP \rightarrow \underset{\substack{(\downarrow \text{CASE}) =_c \text{GEN} \\ (\uparrow \text{POSS}) = \downarrow}}{\boxed{NP}} \qquad N$$

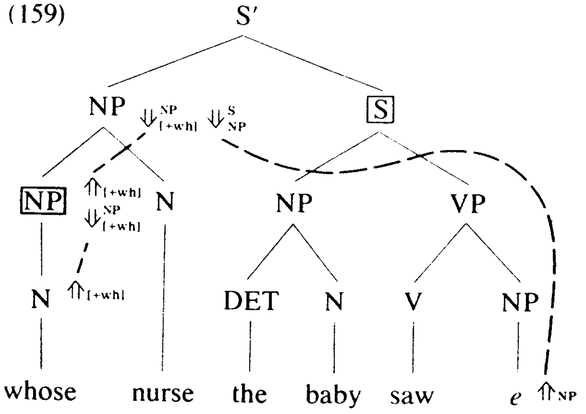
The genitive bounding node also blocks a direct correspondence for the [+wh] metavariables in (155), but a simple schema can be added to rule (157) to circumvent the blocking effect just for interrogative dependencies. This schema, $\Uparrow_{[+wh]} = \Downarrow^{NP}_{[+wh]}$, splits what seems to be a single control domain into two separate domains, one embedded inside the other. It equates a [+wh] controllee for the upper domain with a [+wh] controller for a lower domain:

(158)

$$\begin{array}{lcc} \text{NP} \rightarrow & \boxed{\text{NP}} & \text{N} \\ & (\downarrow \text{CASE}) =_c \text{GEN} & \\ & (\uparrow \text{POSS}) = \downarrow & \\ & \Uparrow_{[+\text{wh}]} = \Downarrow^{\text{NP}}_{[+\text{wh}]} & \end{array}$$

Because this schema links only [+wh] metavariables, constituent control only for interrogatives is possible inside the genitive NP;[26] control for empty NPs is prohibited. The relevant c-structure relations for sentence (155) are illustrated in (159):

(159)



Special constraints have been proposed in transformational theory (e.g., Ross's 1967 Left Branch Condition) to account for the asymmetry in (155) and (156). The lexical-functional description of these facts is stated within the grammar for English, without postulating extragrammatical universal constraints. It thus predicts that this is an area of variation among languages.

In contrast to the nonuniform bounding characteristics of NPs, it can be argued that in languages like English, Icelandic, and French, all Ss are bounding nodes (see the discussions of verb inversion in control domains in Bresnan and Grimshaw 1978 and Zaenen 1980). If so, the Bounding Convention would also block the derivation of sentences such as (160), where the controllee is inside a VP *that*-complement:

(160)
The girl wondered who the nurse claimed that the baby saw ____.

The linking schema appearing in the alternative S' rule (161) will let the dependency go through in this case:

(161)

$$\begin{array}{lcc} \text{S}' \rightarrow & (\textit{that}) & \boxed{\text{S}} \\ & & \uparrow = \downarrow \\ & & \Uparrow = \Downarrow^{\text{S}} \end{array}$$

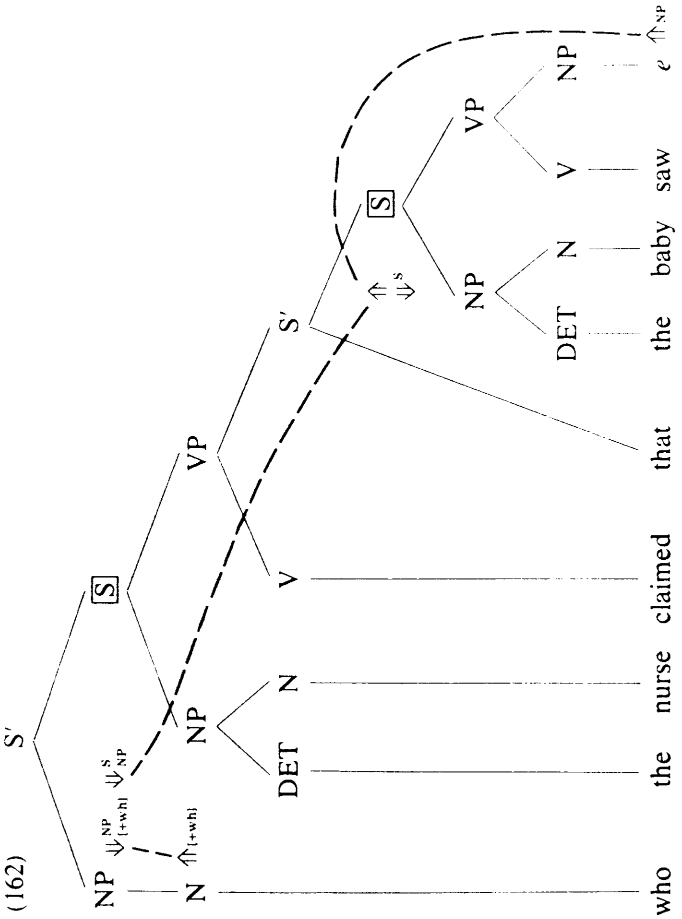
Neither of the metavariables in this linking schema has a categorial subscript. This is an abbreviation for a finite set of alternative schemata of the form $\Uparrow_c = \Downarrow^{\text{S}}_c$, where $c$ is one of the types NP, [+wh], PP, etc. Thus, this schema will link metavariables of any type, passing on to the lower controller the compatibility requirement of the upper one. With this rule, the c-structure (162) is assigned to the sentential complement in (160). Observe that the *that* node belongs to the control domain of the *who* $\Downarrow^{\text{S}}_{\text{NP}}$ controller, since there is no bounding node on the path leading down to it. The $\Uparrow$ on the left of the linking schema is thus instantiated with the $\Downarrow^{\text{S}}_{\text{NP}}$-variable of the *who* node. A separate variable is introduced for the $\Downarrow^{\text{S}}$ on the right, and this is substituted for the $\Uparrow_{\text{NP}}$ of the empty NP, which belongs to the domain rooted in the complement S. The semantically appropriate connections for (160) are thus established.[27]

The definitions of our theory place controllees in a one-to-one correspondence with domain roots and hence with lexical signatures. Our definitions do not establish a correspondence between controllees and arbitrary constituents: there is nothing to prevent control domains from overlapping, and any constituent in several domains may contain several controllees.[28] Control domains will overlap whenever a domain root belonging to the domain of a higher controller is not marked as a bounding node. The potential for multiple dependencies into a single constituent is greater for languages whose grammars specify fewer bounding nodes. The hypothesis that Swedish has fewer bounding nodes than English would thus account for the less restrictive patterns of Swedish dependencies.

There are examples of multiple dependencies in English, however, which we will use to illustrate the operation of our formal mechanism. The recent literature contains many discussions of the interaction of *tough*-movement and questions (see Chomsky 1977b and Fodor 1978, for example, and the references cited therein):[29]

(163)
I wonder which violin the sonata is tough for her to play ____ on ____.

(162)



As we will see, the nodes in the VP′ in this example lie within two control domains, one rooted in the VP′ in the sentential complement of *tough* and the other rooted in the S after *which*. Before exploring the interactions in this sentence, we sketch a grammar for simple *tough*-movement constructions.

A predicate like *tough* is an adjective that can occur as the head of an adjective phrase. Among the alternative expansions for AP is one that allows the adjective to be followed by a sentential complement:

(164)

$$\begin{array}{lll} \mathrm{AP} \rightarrow & \mathrm{A} & \mathrm{S}' \\ & & (\uparrow \textsc{scomp}) = \downarrow \end{array}$$

The VP must of course permit APs as complements to copular verbs, but the details of the VP grammar do not concern us here. *Tough* predicates take infinitival sentential complements, so the category S′ must also have an alternative expansion. Rule (165) allows S′ to expand as a *for*-complementizer followed by a subject NP and a VP′:

(165)

$$\begin{array}{llll} \mathrm{S}' \rightarrow & \mathit{for} & \mathrm{NP} & \mathrm{VP}' \\ & & (\uparrow \textsc{subj}) = \downarrow & \uparrow = \downarrow \\ & & & (\uparrow \textsc{topic}) = \Downarrow^{\mathrm{VP}'}_{\mathrm{NP}} \end{array}$$

The TOPIC schema identifies the TOPIC with an NP controller metavariable whose corresponding controllee must be inside the VP′. Sentences such as (166), where the subject of *tough* has a clause-internal function in an embedded *that*-complement, justify treating this as a constituent control dependency:

(166)
Mary is tough for me to believe that John would ever marry ___.

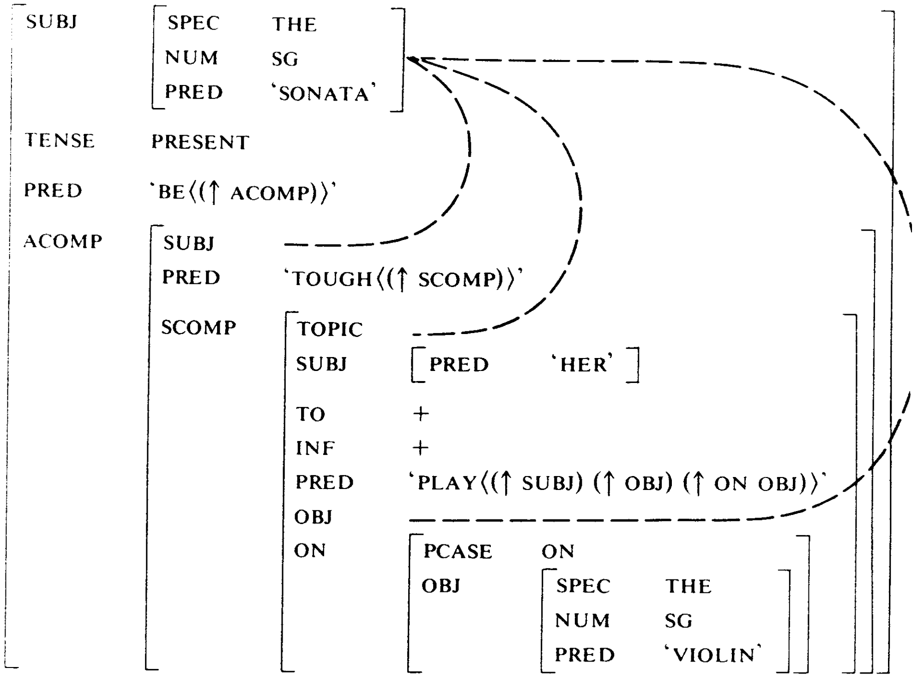
In some respects the TOPIC function is like the FOCUS function introduced earlier for indirect questions. It raises an entity with a clause-internal function to a canonical position in the f-structure hierarchy, providing an alternative access path for various anaphoric rules (cf. note 23). There are substantive differences between TOPIC and FOCUS, however. The FOCUS relation marks new information in the sentence or discourse and therefore is not identified with any other elements. The TOPIC function is a place-holder for old information; its value *must* be linked, either functionally or anaphorically, to some other element. For *tough* predicates, the TOPIC is functionally controlled by a schema in the adjective's lexical entry:[30]

(167)
tough: A, $(\uparrow \text{PRED}) = \text{'TOUGH}\langle(\uparrow \text{SCOMP})\rangle\text{'}$
$(\uparrow \text{SCOMP TOPIC}) = (\uparrow \text{SUBJ})$

With these specifications, the c-structure for the simple *tough*-movement sentence (168) is as shown in (170), and its f-structure is displayed in (169):

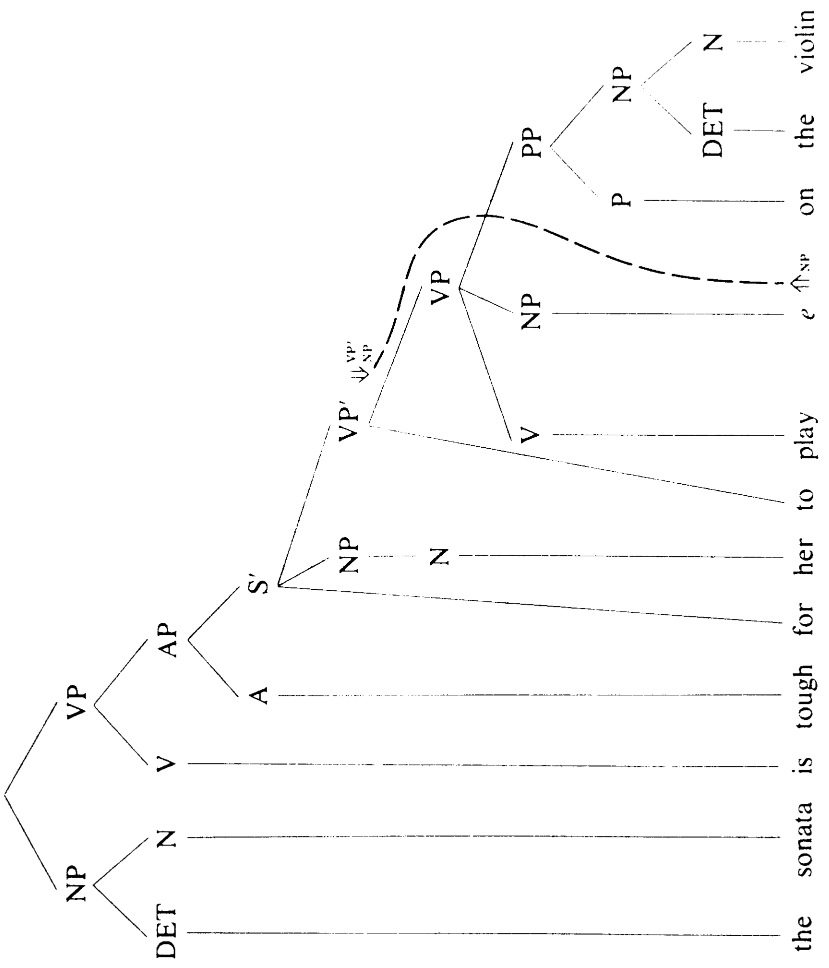
(168)
The sonata is tough for her to play ___ on the violin.

(169)

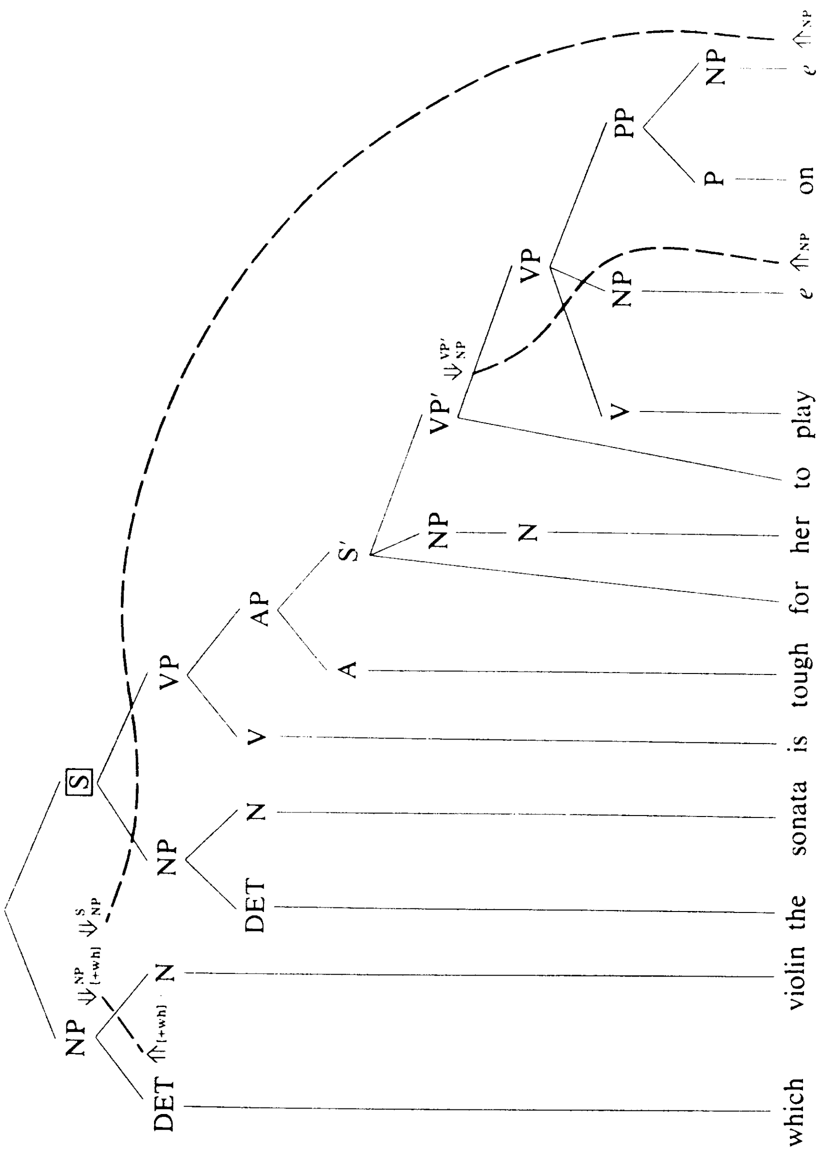


We are now ready to examine the double dependency in (163). In this sentence, *violin* has become the FOCUS of an indirect question. The c-structure for the complement of *wonder* is shown in (171). Since the VP′ domain is introduced without a bounding node, there is nothing to block the correspondence between the object NP of *on* and the NP

(170)



(171)



controller for *which violin*. The correspondence for the TOPIC metavariables in *tough*'s complement is established just as in the simpler example above. Thus, the metavariables can be properly instantiated, and the intuitively correct f-structure will be assigned to this sentence.

As has frequently been noted, the acceptability of these double dependencies is sensitive to the relative order of controllees and controllers. If *sonata* is questioned and *violin* is the *tough* subject, the result is the ungrammatical string (172):

(172)
*I wonder which sonata the violin is tough for her to play ___ on ___.

The reading of this sentence in which *which sonata* is the object of *on* and *violin* is the object of *play* is semantically unacceptable, but the semantically well-formed reading of our previous example (163) is not available. Similarly, Bach 1977 observes that potentially ambiguous sentences are rendered unambiguous in these constructions. Sentence (173) can be assigned only the reading in which *doctor* is understood as the object of *to* and *patient* is the object of *about*, even though the alternative interpretation is equally plausible:

(173)
Which patient is that doctor easiest to talk to ___ about ___?

As Baker 1977, Fodor 1978, and others have pointed out, there is a simple and intuitive way of characterizing the acceptable dependencies in these examples. If a line is drawn from each gap to the various lexical items that are candidates for filling it, then the permissible dependencies are just those in which the lines for the separate gaps do not cross. Or, to use Fodor's terminology, only nested dependencies seem to be allowed.

The nested pattern of acceptable dependencies is an empirical consequence of the requirement (147e) that corresponding metavariables be nearly nested. However, this restriction in our definition of proper instantiation is strongly motivated by independent theoretical considerations: as we point out in section 4.8, this requirement provides a sufficient condition for proving that lexical-functional languages are included within the set of context-sensitive languages. Thus, our restriction offers not only a description of the observed facts, but also a formal basis for explaining them.

As the first step in formalizing the notion of a nearly nested correspondence, we establish an ordering on the bounded domination metavariables attached to a c-structure. We order the c-structure nodes so that each node comes before its daughters and right-sister (if any), and all its daughters precede its right-sister. If the node that one metavariable is attached to precedes another metavariable's node, then we say that the first metavariable precedes the second. The ordering of metavariables can be described more perspicuously in terms of a labeled bracket representation of the c-structure tree. If metavariables are associated with the *open* brackets for the nodes they are attached to, then the left-to-right sequence in the labeled bracketing defines the metavariable ordering. This is illustrated with the (partial) bracketing for sentence (163) shown in part (a) of figure 4.1. We see from this representation that the $\Downarrow^{S}_{NP}$ on the fronted NP is ordered before the $\Downarrow^{VP'}_{NP}$ and that *play*'s direct object $\Uparrow_{NP}$ is ordered before the controllee after *on*.

Drawing lines between corresponding metavariables as ordered in part (a) of figure 4.1 illustrates the intuitive contrast between nested and crossed dependencies. The lines are shown in part (b) of figure 4.1 for the acceptable nested reading of (163) and in part (c) for the unacceptable crossed dependency. A precise formulation of this intuitive distinction can be given in terms of the definition of a crossed correspondence:

(174)
*Definition of Crossed Correspondence*
The correspondence of two metavariables $m_1$ and $m_2$ is *crossed* by a controller or controllee $m_3$ if and only if all three variables have compatible categorial subscripts and $m_3$ but not its corresponding controllee or controller is ordered between $m_1$ and $m_2$.

Obviously, a correspondence is nested if and only if it is not crossed. All the correspondences in the acceptable readings for the examples above are nested according to this definition, but the correspondences in the unacceptable readings are not.

Metavariable correspondences can be allowed limited departures from strict nesting without undermining the context-sensitivity of lexical-functional languages. We associate with each metavariable correspondence an integer called its *crossing degree*. This is simply the number of controllers and controllees by which that correspondence is crossed. A correspondence is strictly nested if its crossing degree is zero. Further, for each lexical-functional grammar we determine an-

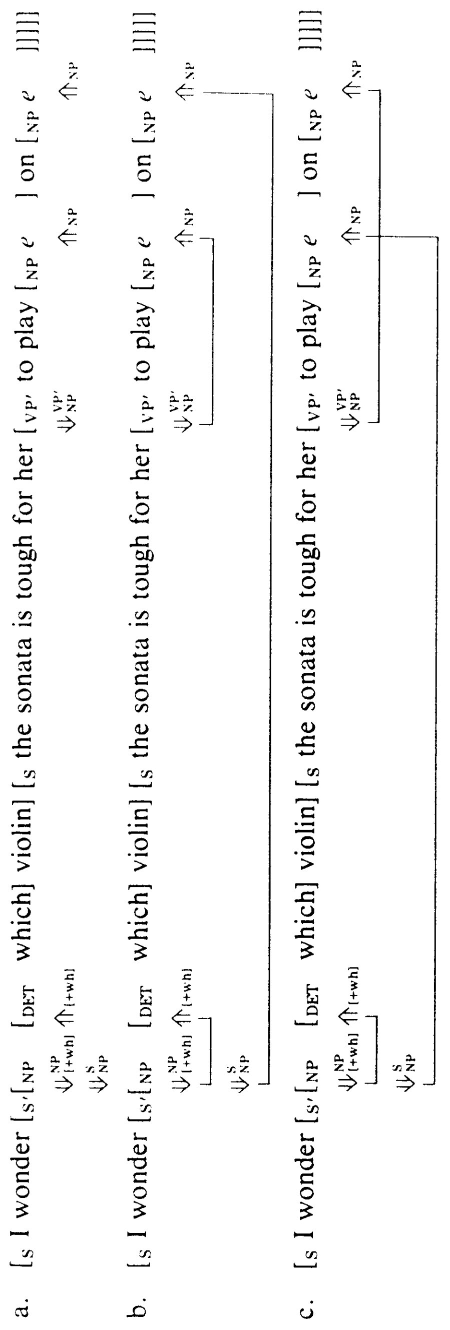

a. [S I wonder [S' [NP [DET which] violin] [S the sonata is tough for her [VP' to play [NP e ] on [NP e ]]]]]
$\Downarrow^{NP}_{[+wh]}$ $\Uparrow_{[+wh]}$ $\Downarrow^{S}_{NP}$ $\Downarrow^{VP'}_{NP}$ $\Uparrow_{NP}$ $\Uparrow_{NP}$
b. [S I wonder [S' [NP [DET which] violin] [S the sonata is tough for her [VP' to play [NP e ] on [NP e ]]]]]
$\Downarrow^{NP}_{[+wh]}$ $\Uparrow_{[+wh]}$ $\Downarrow^{S}_{NP}$ $\Downarrow^{VP'}_{NP}$ $\Uparrow_{NP}$ $\Uparrow_{NP}$
c. [S I wonder [S' [NP [DET which] violin] [S the sonata is tough for her [VP' to play [NP e ] on [NP e ]]]]]
$\Downarrow^{NP}_{[+wh]}$ $\Uparrow_{[+wh]}$ $\Downarrow^{S}_{NP}$ $\Downarrow^{VP'}_{NP}$ $\Uparrow_{NP}$ $\Uparrow_{NP}$

**Figure 4.1**

other number, the *crossing limit* of the grammar. A nearly nested correspondence is then defined as follows:

(175)
*Definition of Nearly Nested Correspondence*
A metavariable correspondence is *nearly nested* if its crossing degree does not exceed the grammar's crossing limit.

The significant formal implication of this definition and the nearly nested restriction on proper instantiation is that for any string the degree of departure from strict nesting is bounded by a constant that is independent of the length of that string.

The examples above suggest that the crossing limit for English is zero. This limit can be maintained even in the face of apparent counterexamples to the nesting proposals of other theories. Since our definition of crossed correspondence (174) only involves metavariables with compatible categorial subscripts, we have no difficulty with acceptable sentences containing crossed dependencies of different categories. Other classes of counterexamples involve interactions of functional and constituent control, but our restrictions are imposed only for constituent control dependencies. Thus, there is no real cross-over in sentences such as (176):

(176)
How nice a man would John be ___ to marry ___?

The *man* NP is linked to the first gap, while *John* is linked to the second. In our theory there is a functional identification between *John*, the SUBJ of the complex predicate *how nice a man*, and the TOPIC of its SCOMP. The controller for the second dependency is thus ordered *after* the first gap. Icelandic stands in contrast to English in having constituent control dependencies that can be described correctly only on the hypothesis that the crossing limit for that language is one (Maling and Zaenen to appear).

We have presented in this section the major formal mechanisms for characterizing the long-distance dependencies of natural language. We have motivated and illustrated our formal apparatus with simple and plausible fragments of English grammar. Constituent control is a syntactic phenomenon of considerable complexity, and there are many empirical and theoretical issues that we have not touched on and some that are still to be resolved. No doubt future research in this area will lead to both substantive and formal refinements of our theory. However, we expect the broad outline of our approach to remain unchanged: lexical-functional grammar treats long-distance dependencies as part of the procedure for producing properly instantiated f-descriptions. These dependencies are governed by c-structure configurations and are not directly sensitive to the f-structures that are ultimately constructed.

## 4.8 Generative Power

We have seen that lexical-functional grammar offers considerable expressive power for describing linguistic phenomena. In this section we examine the position of LFG in the Chomsky hierarchy of generative capacity. The most important result is that our formal system, with two well-motivated restrictions on c-structure derivations that we discuss below, is *not* as powerful as a general rewriting system or Turing machine. In fact, lexical-functional languages are included within the class of context-sensitive languages. On the lower end of the scale, we show that LFG has greater generative power than the class of context-free grammars.

For a string to be a member of the language generated by a lexical-functional grammar, it must satisfy five requirements:

(177)
a. It must be the terminal string of a valid c-structure derivation.
b. There must be a properly instantiated f-description associated with that derivation.
c. The f-description must be consistent and determinate, with a unique minimal solution.
d. The minimal f-structure solution must satisfy all constraints in the f-description.
e. The f-structure must be complete and coherent.

Given a single c-structure derivation for a string of length *n* (a tree to whose nodes the appropriate functional schemata are attached), there are finite procedures for deciding whether or not (177b)–(177e) hold. Determining proper instantiation for immediate domination metavariables is trivial. Since the given tree has only a finite number of finite control domains, it is also computable whether or not the bounded domination metavariables are properly instantiated. The instantiated f-description has a finite number of statements in it, so the algorithm outlined in section 4.4 and in the appendix produces its unique minimal solution, if it is consistent and determinate. Evaluating a constraining statement requires only a finite traversal of the f-structure,[31] and the Completeness and Coherence Conditions can similarly be checked by a finite computation on the f-structure.

Thus, all that is needed to prove that the grammaticality of any string is decidable is a terminating procedure for enumerating all possible c-structures for the string, so that the functional correctness of each one can then be verified. C-structures are generated by context-free grammars, and there are well-known decision procedures for the membership problem of grammars in this class. That is, there exist algorithms for determining whether there is *at least one* way of deriving the string. Deciding that a string is derivable, however, is not the same as enumerating for inspection all of its derivations. Indeed, there are grammars for which neither the number of derivations that a given string might have nor the number of nodes in a single derivation is bounded. While it may be determined that such a string has one derivation and thus belongs to the language of the c-structure grammar, there is no way of deciding whether or not there exists among all of its derivations one that satisfies the functional requirements of our theory. Suppose that at some point we have examined all derivations with less than *m* nodes and found them all to be functionally deviant. This does not mean that all derivations with *m* + 1 nodes will also be unsatisfactory. Since this can be true for any *m*, the grammaticality of that string cannot be decided in a finite number of steps. (This difficulty arises not just with our formalism but with any system in which the definition of grammaticality involves an evaluation or interpretation of the context-free derivations.)

A context-free grammar can produce an unbounded number of derivations of arbitrary size for a string, either because its rules permit a single category to appear twice in a nonbranching chain, or because expansions involving the empty string are not sufficiently restricted. The rules in (178) illustrate the first situation:

(178)
X → Y
Y → Z
Z → X

Any string which has a derivation including the category X will be infinitely ambiguous. There is a larger derivation with the domination chain X–Y–Z–X replacing the single X, and a still larger one with one of those Xs replaced by another chain, and so on. The derivations that result from rules of this sort are in a certain sense peculiar. The nonbranching recursive cycles permit a superstructure of arbitrary size to be constructed over a single terminal or group of terminals (or even over the empty string). The c-structure is thus highly repetitive, and the f-description, which is based on a fixed set of lexical schemata and arbitrary repetitions of a finite set of grammatical schemata, is also. While the c-structure and f-structure can be of unbounded size, they encode only a finite amount of nonredundant information that is relevant to the functional or semantic interpretation of the string.

Such vacuously repetitive structures are without intuitive or empirical motivation. Presumably, neither linguists nor language learners would postulate rules of grammar whose purpose is to produce these derivations. However, linguists and language learners both are likely to propose rules whose purpose is to express certain surface structure generalizations but which have derivations of this sort as unintended consequences. For example, suppose that the grammar that includes (178) also has a large number of alternative rules for expanding Y and Z. Suppose further that except for the undesired cyclic X–Y–Z–X chain, X can dominate everything that Y and Z dominate. Only the intended derivations are permitted if X expands to a new category Y′ whose rules are exactly the same as the rules for Y except that another new category Z′ appears in place of Z in (178). The rules for Z′ are those of Z without the X alternative. This much more complicated grammar does not make explicit the almost complete equivalence of the Y–Y′ and Z–Z′ categories. Except for the one spurious derivation, the original grammar (178) is a much more revealing description of the linguistic facts.

The following rules illustrate how derivations of arbitrary size may also result from unrestricted empty string expansions:

(179)
P → P P
P → *e*

If a P dominates (either directly or indirectly) a lexical item in one derivation, there will be another derivation in which that P has a mother and sister which are both P, with the sister expanding to the empty string. Without further stipulations, rules of this sort can apply an indefinite number of times. We introduced empty strings in section 4.7 to represent the lower end of long-distance dependencies. These *e*'s have controllee metavariables and thus are uniquely associated with the lexical signature of a control domain. The possibility of arbitrary repetitions does not arise because derivations for a string of length *n* can have no more than *n* controlled *e*'s. An empty string may appear in a c-structure rule for another reason, however. It can alternate with other rule elements in order to mark them as optional. An *optionality e* is a generalization of the standard parenthesis notation for c-structure optionality; it permits functional schemata to be introduced when the optional constituents are omitted. An optionality *e* does not have the controllee metavariable that inhibits repetitions of controlled *e*'s and, according to the standard interpretation of context-free rules, may appear in derivations indefinitely many times with no intervening lexical items. These derivations are redundant and unmotivated, just like those with nonbranching dominance cycles. The possibility of repeating rule elements with fixed schema sets and no new lexical information is, again, an unintended consequence of a simple notational device for conflating sets of closely related rules.

Having argued that the vacuous derivations involving nonbranching dominance chains and repeated optionality *e*'s are unmotivated and undesired, we now simply exclude them from functional consideration. We do this by restricting what it means to be a "valid" c-structure derivation in the sense of (177a):

(180)
*Definition of Valid Derivation*
A c-structure derivation is valid if and only if no category appears twice in a nonbranching dominance chain, no nonterminal exhaustively dominates an optionality *e*, and at least one lexical item or controlled *e* appears between two optionality *e*'s derived by the same rule element.

This definition, together with the fact that controlled *e*'s are associated with unique lexical signatures, implies that for any string the size and number of c-structure derivations relevant to our notion of grammaticality is bounded as a function of *n*, even though no such bounds exist according to the standard interpretation for context-free grammars. Note that this restriction on derivations does not affect the language of the c-structure grammar: it is well known that a string has a valid c-structure with no cycles and no *e*'s if and only if it has any c-structure at all (see Hopcroft and Ullman 1969).

With the validity of a derivation defined as in (180), the following theorem can be proved:

(181)
*Decidability Theorem*
For any lexical-functional grammar *G* and for any string *s*, it is decidable whether *s* belongs to the language of *G*.

We observe that algorithms exist for enumerating just the finite number of valid derivations, if any, that *G* assigns to *s*. A conventional context-free parsing algorithm, for example, can easily be modified to notice and avoid nonbranching cycles, to keep track of the source of optionality *e*'s and avoid repetitions, and to postulate no more controlled *e*'s than there are words in the string. With the valid derivations in hand, there are algorithms, as outlined above, for determining whether any of them satisfies the functional conditions (177b)–(177e). Theorem (181) is thus established.[32]

Theorem (181) sets an upper bound on the generative capacity of lexical-functional grammar: only the recursive as opposed to recursively enumerable languages are generable. It is possible to set a tighter bound on the generative power of our formalism. Because of the nearly nested restriction on proper instantiation, for any lexical-functional grammar *G* a nondeterministic linear bounded automaton can be constructed that accepts exactly the language of *G*. Lexical-functional languages are therefore included within the context-sensitive languages. The details of this construction are quite complicated and will be presented in a separate publication. In brief, the c-structure with attached schemata for any string of length *n* can be discovered and represented by an automaton with a working tape whose size is bounded by a linear function of *n*. This automaton, however, cannot introduce actual variables and substitute them for metavariables as the instantiation procedure specifies, since that would require a nonlinear amount of space (roughly proportional to *n* log *n*). Instead, it uses the arrangement of metavariables in the c-structure to determine the implicit synonymy relations that the actual variables would simply make explicit. The nearly nested restriction guarantees that these relations can be computed using a linear amount of working storage.[33] With synonymous metavariables identified, the functional well-formedness conditions (177c)–(177e) can also be verified in a linear amount of space.

The generative power of lexical-functional grammar is obviously bounded from below by the class of context-free grammars. Any given context-free grammar is a legitimate c-structure grammar with no grammatical schemata. As noted above, the strings with valid c-structure derivations are exactly those that belong to the context-free language. The sets of schemata for those derivations are empty and are vacuously instantiated to produce an empty f-description whose unique minimal solution is the null f-structure. The functional component thus does no filtering, and the c-structure grammar under our interpretation is weakly equivalent to the grammar interpreted in the ordinary context-free way.

In fact, LFG has greater generative power than the class of context-free grammars, for it allows grammars for languages that are known not to be context-free. The language $a^n b^n c^n$ is a classic example of such a language. Its strings consist of a sequence of *a*'s followed by the same number of *b*'s and then *c*'s. A grammar for this language is shown in (182):

(182)

$$\mathrm{S} \rightarrow \underset{\uparrow=\downarrow}{\mathrm{A}}\quad \underset{\uparrow=\downarrow}{\mathrm{B}}\quad \underset{\uparrow=\downarrow}{\mathrm{C}}$$

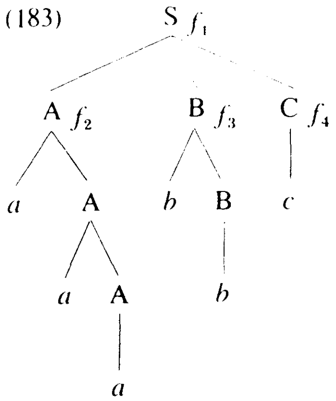
$$\mathrm{A} \rightarrow \left\{ \begin{array}{c} \underset{(\uparrow\ \textsc{count})=0}{a} \\ \\ a \quad \underset{(\uparrow\ \textsc{count})=\downarrow}{\mathrm{A}} \end{array} \right\}$$

$$\mathrm{B} \rightarrow \left\{ \begin{array}{c} \underset{(\uparrow\ \textsc{count})=0}{b} \\ \\ b \quad \underset{(\uparrow\ \textsc{count})=\downarrow}{\mathrm{B}} \end{array} \right\}$$

$$\mathrm{C} \rightarrow \left\{ \begin{array}{c} \underset{(\uparrow\ \textsc{count})=0}{c} \\ \\ c \quad \underset{(\uparrow\ \textsc{count})=\downarrow}{\mathrm{C}} \end{array} \right\}$$

The c-structure rules produce *a*'s, *b*'s, and *c*'s in sequences of arbitrary length, as illustrated by the c-structure for *aaabbc* in (183):

(183)



The lengths of those sequences, however, are encoded in the f-structure. For each of the A, B, and C nodes in the tree, the number of COUNT attributes in the ↓ f-structure of that node is a count of the elements in that node's terminal sequence. Thus, the f-structures shown in (184) for the $f_2$, $f_3$, and $f_4$ nodes have three, two, and one COUNTs, respectively.

(184)

$$_{f_2}\left[\textsc{count}\quad \left[\textsc{count}\quad \left[\textsc{count}\quad 0\right]\right]\right]$$

$$_{f_3}\left[\textsc{count}\quad \left[\textsc{count}\quad 0\right]\right]$$

$$_{f_4}\left[\textsc{count}\quad 0\right]$$

The attempt to equate these three f-structures in accordance with the schemata on the S rule leads to a violation of the Uniqueness Condition, and the string is marked ungrammatical. Only if the terminal sequences are all of the same length can the f-structures be combined.

The f-structure in this grammar records the one string property, sequence length, that is crucially needed for this particular context-sensitive test. If instead we let the f-structure be a complete, isomorphic image of the c-structure tree, we can describe a repetition language, another classical example of a non-context-free language. This is a language whose sentences are all of the form $\omega\omega$, where $\omega$ stands for an arbitrary string over some vocabulary. We start with a simple context-free grammar for the strings $\omega$, for example, the rule in (185a).

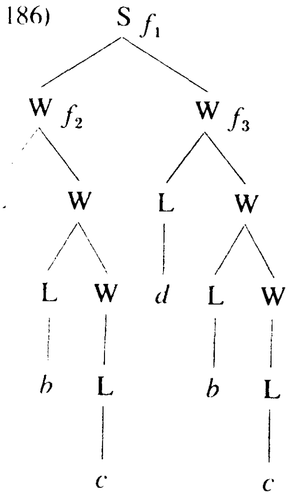
(185)

a. $W \rightarrow L \begin{pmatrix} W \\ (\uparrow W)=\downarrow \end{pmatrix}$

b. $S \rightarrow \underset{\uparrow=\downarrow}{W} \quad \underset{\uparrow=\downarrow}{W}$

All words in the vocabulary are assumed to belong to the lexical category L, so this rule generates arbitrary strings under right-branching tree structures. If for every word *x* there is a distinct symbol x, and if *x* has (↑ L) = x as its only lexical schema, the ↓ f-structure of a W node will be an exact image of its subtree. For example, (186) shows the c-structure that this grammar would assign to the ungrammatical string *bcdbc*, and (187) gives the f-structures for the two topmost W nodes:

(186)



(187)

$$\begin{bmatrix} \text{L} & \text{A} \\ \text{W} & \begin{bmatrix} \text{L} & \text{B} \\ \text{W} & \begin{bmatrix} \text{L} & \text{C} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{L} & \text{D} \\ \text{W} & \begin{bmatrix} \text{L} & \text{B} \\ \text{W} & \begin{bmatrix} \text{L} & \text{C} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

These f-structures contradict the schemata on the S rule, which assert that they are identical. The f-structures for the two Ws in sequence will

be the same only if their subtrees and hence their terminal strings are also the same.

We can thus characterize within our formalism at least some of the non-context-free context-sensitive languages. There is nothing devious or obscure about the grammars for these languages: they use ordinary functional mechanisms in perfectly straightforward ways. The additional generative power comes from two features of LFG, functional composition and the equality predicate. Function composition permits f-structures to encode a wide range of tree properties, while the equality predicate can enforce a match between the properties encoded from different nodes. We can be even more specific about the source of our context-sensitive power. If all schemata in a grammar equate attribute values only to constants (e.g., schemata of the form $d_1 = d_2$, where $d_2$ designates a symbol or semantic form), then a weakly equivalent context-free grammar can be constructed. In this grammar the information contained in the f-structure is encoded in an enlarged set of context-free categories. The additional power of lexical-functional grammar stems from schemata that equate two f-structures, for example, the identification schemata in the examples above.

We have shown that lexical-functional languages properly include the context-free languages and are included within the context-sensitive languages. LFG's generative capacity is both a strong point of our theory and also something of an embarrassment. Huybregts 1976 has argued that dependencies of the sort illustrated by (185) are quite productive in Dutch,[34] and such phenomena have been claimed to exist in other languages as well (e.g., Mohawk (Postal 1964) and the English *respectively* construction). Mechanisms of this power must therefore be a part of any adequate theory of human language.

On the other hand, the problem of recognizing languages with context sensitivities can be computationally much more complex than the recognition problem for context-free languages. If our system turns out to have full context-sensitive power, then there are no known solutions to the recognition problem that require less than exponential computational resources in the worst case. It might therefore seem that, contrary to the Competence Hypothesis, lexical-functional grammars cannot be naturally incorporated into performance models that simulate the apparent ease of human comprehension.

There are several reasons why this conclusion does not necessarily follow. First, an explanatory linguistic theory undoubtedly will impose a variety of substantive constraints on how our formal devices may be employed in grammars of human languages. Some candidate constraints have been mentioned in passing (e.g., the constraints on functional control schemata and the principle of functional locality), and others are under current investigation. It is quite possible that the worst case computational complexity for the subset of lexical-functional grammars that conform to such constraints will be plausibly subexponentional. Second, while the Competence Hypothesis asserts that a grammar will be a significant component of a performance model, the grammar is not identified with the processor that interprets it. An adequate theory of performance might impose certain space and time limitations on the processor's capabilities or specify certain nongrammatical heuristic strategies to guide the processor's computations (see for example the scheduling heuristics described in chapter 11). Given these further assumptions, the performance model might actually exhibit the worst case behavior very rarely and then only under special circumstances. Finally, it is quite possible that the exponential explosion is in fact psychologically realistic. For our formal system, this processing complexity is not the result of a lengthy search along erroneous paths of computation. Rather, it comes about only when the c-structure grammar assigns an exponential number of c-structure ambiguities to a string. To the extent that c-structure is a psychologically real level of representation, it seems plausible that ambiguities at that level will be associated with increased cognitive load.

We conjecture, then, that the generative power of our system is not only necessary for adequate linguistic descriptions but is also compatible with realistic models of psycholinguistic performance. In keeping with the Competence Hypothesis, we believe that performance models that incorporate linguistically justified lexical-functional grammars will ultimately provide an explanatory account of the mental operations that underlie human linguistic abilities.

## Appendix: F-Description Solution Operators

An intuitive description of our f-description solution algorithm was presented in section 4.4. The algorithm involves three basic operators: Locate, Merge, and Include. If $d_1$ and $d_2$ are designators, then an f-description equality of the form $d_1 = d_2$ is processed by performing Merge[Locate[$d_1$], Locate[$d_2$]], and a membership statement of the form $d_1 \in d_2$ is processed by performing Include[Locate[$d_1$], Locate[$d_2$]]. We now give the formal definitions of these operators.

Locate, Merge, and Include all cause modifications to a collection of entities and variable assignments $C$, either by modifying an already existing entity or by substituting one entity for every occurrence of another. We specify substitution as a separate suboperator, since it is common to all three operators:

(188)
*Definition of Substitute*
For two entities *old* and *new*, Substitute[*new*, *old*] replaces all occurrences of *old* in $C$ with *new*, assigns *new* as the value of variables that previously had *old* as their assignment (in addition to any variables that had *new* as their value previously), and removes *old* from $C$.

Applying the Substitute operator makes all previous designators of *old* and *new* be designators of *new*.

The Locate operator takes a designator $d$ as input. If successful, it finds a value for $d$ in a possibly modified entity collection.

(189)
*Definition of Locate*

a. If $d$ is an entity in $C$, then Locate[$d$] is simply $d$.
b. If $d$ is a symbol or semantic form character string, Locate[$d$] is the symbol or semantic form with that representation.
c. If $d$ is a variable,
    If $d$ is already assigned a value in $C$, Locate[$d$] is that value.
    Otherwise, a new place-holder is added to $C$ and assigned as the value of $d$. Locate[$d$] is that new place-holder.
d. Otherwise, $d$ is a function-application expression of the form $(f\ s)$.
  Let $F$ and $S$ be the entities Locate[$f$] and Locate[$s$], respectively.
    If $S$ is not a symbol or place-holder, or if $F$ is not an f-structure or place-holder, the f-description has no solution.
    If $F$ is an f-structure:
      If $S$ is a symbol or place-holder with a value defined in $F$, then Locate[$d$] is that value.
      Otherwise, $S$ is a place-holder or a symbol for which $F$ has no value. $F$ is modified to define a new place-holder as the value of $S$. Locate[$d$] is that place-holder.
    Otherwise, $F$ is a place-holder. A new f-structure $F'$ is constructed with a single pair that assigns a new place-holder value to $S$, and Substitute[$F'$, $F$] is performed. Locate[$d$] is then the new place-holder value.

(189a) provides closure by allowing an entity to serve as a designator of itself. The recursive invocations of Locate that yield $F$ and $S$ in (189d) enable the values of all functional compositions to be obtained. The consistency check is specified in the first clause of (189d). A Locate attempt fails if it requires an entity already known not to be an f-structure to be applied as a function, or an entity known not to be a symbol to be used as an argument.

The Merge operator is also defined recursively. It takes two entities $e_1$ and $e_2$ as input. Its result is an entity $e$, which might be newly constructed. The new entity is substituted for both $e_1$ and $e_2$ in $C$ so that all designators of $e_1$ and $e_2$ become designators of $e$ instead.

(190)
*Definition of Merge*

a. If $e_1$ and $e_2$ are the same entity, then Merge[$e_1$, $e_2$] is that entity and $C$ is not modified.

b. If $e_1$ and $e_2$ are both symbols or both semantic forms, the f-description has no solution.

c. If $e_1$ and $e_2$ are both f-structures, let $A_1$ and $A_2$ be the sets of attributes of $e_1$ and $e_2$, respectively. Then a new f-structure $e$ is constructed with

$$e = \{\langle a, v\rangle \mid a \in A_1 \cup A_2 \text{ and } v = \text{Merge[Locate[}(e_1\ a)\text{], Locate[}(e_2\ a)\text{]]}\}.$$

Substitute[$e$, $e_1$] and Substitute[$e$, $e_2$] are both performed, and the result of Merge[$e_1$, $e_2$] is then $e$.

d. If $e_1$ and $e_2$ are both sets, then a new set $e = e_1 \cup e_2$ is constructed. Substitute[$e$, $e_1$] and Substitute[$e$, $e_2$] are both performed, and the result of Merge[$e_1$, $e_2$] is then $e$.

e. If $e_1$ is a place-holder, then Substitute[$e_2$, $e_1$] is performed and the result of Merge[$e_1$, $e_2$] is $e_2$.

f. If $e_2$ is a place-holder, then Substitute[$e_1$, $e_2$] is performed and the result of Merge[$e_1$, $e_2$] is $e_1$.

g. Otherwise, $e_1$ and $e_2$ are entities of different types, and the f-description has no solution.

The consistency check in (190b) ensures that nonidentical symbols and semantic forms are not combined, and the checks in (190c,d) guarantee that entities of different known types (i.e., excluding place-holders) cannot be merged. The recursion in (190c) propagates these checks to all the substructures of two f-structures, building compatible values for common function names as it proceeds down level by level until it reaches non-f-structure values.[35]

The Include operator has a particularly simple specification in terms of the Merge operator. It takes two entities $e$ and $s$ as input and is defined as follows:

(191)
*Definition of Include*
Perform Merge [$\{e\}$, $s$].

The first entity given to Merge is a new set with $e$ as its only member. The set-relevant clauses of the Merge definition are thus applicable: if $s$ is also a set, for example, (190d) indicates how its other elements will be combined with $e$.

With these operator definitions, the fundamental theorem that our algorithm produces solutions for all and only consistent f-descriptions can easily be proved by induction on the number of statements in the f-description. Suppose an entity collection $C$ is a solution for an f-description of $n - 1$ statements. Then the collection after successfully performing Merge[Locate[$d_1$], Locate[$d_2$]] is a solution for the description formed by adding $d_1 = d_2$ as an $n$th statement, and the collection after successfully performing Include[Locate[$d_1$], Locate[$d_2$]] is a solution for the description formed by adding $d_1 \in d_2$ as an $n$th statement. If the Locate, Merge, or Include operators fail, the larger f-description is inconsistent and has no solution at all.

## Notes

We are indebted to Martin Kay, Beau Sheil, and Tom Wasow for valuable suggestions and fruitful discussions over a period of several years. We also wish to thank Ken Church, Elisabet Engdahl, Kris Halvorsen, and Annie Zaenen for commenting on earlier drafts of this chapter. This work was supported in part by a grant from the Alfred P. Sloan Foundation for research in cognitive science at the Massachusetts Institute of Technology.

1. Kaplan 1975a,b gives an early version of the Competence Hypothesis and discusses some ways in which the grammatical and processing components might interact. Also see chapter 11 of this volume.

2. Semantic forms with a lexical source are often called *lexical forms*. Less commonly, semantic forms are produced by syntactic rules, for example, to represent unexpressed pronouns; this will be illustrated in section 4.6 in the discussion of English imperative subjects.

3. This chapter is not concerned with the details of the semantic translation procedure for NPs, and the specifications for the SPEC and common noun PRED features are simplified accordingly. With more elaborate expressions for these features, NPs can also be translated into a higher-order intensional logic by a general substitution procedure. For instance, suppose that the symbol A is taken as an abbreviation for the semantic form '$\lambda Q \lambda P \exists x \langle Q\langle x \rangle \wedge P\langle x \rangle \rangle$', which represents the meaning of an existential quantifier, and suppose that 'GIRL' is replaced by the expression '$(\uparrow$ SPEC$)\langle$GIRL$'\rangle$'. Then the translation for the SUBJ f-structure would be a formula in which the quantifier is applied to the common noun meaning. See Halvorsen forthcoming for an extensive discussion of f-structure translation and interpretation.

4. This correlation of rule properties is a significant difference between lexical-functional grammar and Relational Grammar (see for example the papers in Perlmutter in press). The two approaches are similar, however, in the emphasis they place on grammatical relations. Bell 1980 offers a more extensive comparison of the two theories.

5. There is an equivalent formulation in which the grammatical relation symbols SUBJ, OBJ, etc., are taken to be the names of functions that apply to f-structure arguments. We would then write SUBJ($f_1$) instead of $f_1$(SUBJ), and the left- and righthand elements of all our expressions would be systematically interchanged. Even with this alternative, however, there are still cases where the function is an unknown (see for example the discussion below of oblique objects). The conceptual consideration underlying our decision to treat f-structures as the formal functions is that only total, finite functions are then involved in the characterization of particular sentences. Otherwise, our conceptual framework would be populated with functions on infinite domains, when only their restriction to the sentence at hand would ever be grammatically relevant. Only this intuition would be affected if the alternative formulation were adopted.

6. Another convention for lexical insertion is to attach the schemata directly to the terminal nodes. While the same functional relationships can be stated with either convention, this alternative requires additional identification schemata in the common case where the preterminal category does not correspond to a distinct functional unit. It is thus more cumbersome to work with.

7. If a schema containing $\uparrow$ is attached to a node whose mother has no $\downarrow$-variable, the $\uparrow$ cannot be properly instantiated and the string is marked ungrammatical. This situation is not likely to occur with immediate domination metavariables but provides an important well-formedness condition for bounded domination. This is discussed in section 4.7.

8. In effect, the instantiation procedure adds to the schemata information about the tree configurations in which they appear. As shown in section 4.4, the f-structure for the sentence can then be inferred without further reference to the c-structure. An equivalent inference procedure can be defined that does not require the introduction of variables and instead takes into account the relative position of schemata in the tree. This alternative procedure searches the c-structure to obtain the information that we are encoding by variables in instantiated schemata. It essentially intermixes our instantiation operations among its other inferences and is thus more difficult to describe.

9. An attribute in an f-structure is thus a special kind of designator, and the notion of a designator's value generalizes our use of the term *value*, which previously referred only to the entity paired with an attribute in an f-structure.

10. This algorithm is designed to demonstrate that the various conditions imposed by our theory are formally decidable. It is unlikely that this particular algorithm will be incorporated intact into a psychologically plausible model of language performance or even into a computationally efficient parser or generator. For these other purposes, functional operations will presumably be interleaved with c-structure computations, and functional data representations will be chosen so as to minimize the combinatoric interactions with the nondeterministic uncertainty of the c-structure rules.

11. Our c-structure rules thus diverge from a strict context-free formalism. We permit the righthand sides of these rules to be regular expressions as in a recursive transition network, not just simply-ordered category sequences. The * is therefore not interpreted as an abbreviation for an infinite number of phrase structure rules. As our theory evolves, we might incorporate other modifications to the c-structure formalism. For example, in a formalism which, although oriented toward systemic grammar descriptions, is closely related to ours, Kay 1979 uses patterns of partially ordered grammatical relations to map between a linear string and his equivalent to an f-structure. Such partial orderings might be particularly well-suited for free word-order languages.

12. The case-marking entry is distinct from the entry for *to* when it serves as a predicate in its own right, as in prepositional complements or adjuncts.

13. Our general Uniqueness Condition is also the most crucial of several differences between lexical-functional grammar and its augmented transition network precursor. ATN SETR operations can arbitrarily modify the f-structure values (or "register contents," in ATN terminology) as they are executed in a left-to-right scan of a rule or network. The register SUBJ can have one value at one point in a rule and a completely different value at a subsequent point. This revision of value assignments is not allowed in LFG. Equations at one point cannot override equations instantiated elsewhere—all equations must be simultaneously satisfied by the values in a single f-structure. As we have seen, the properties of that f-structure thus do not depend on the particular sequence of steps by which schemata are instantiated or the f-description is solved.

14. In a more detailed treatment of morphology, the schemata for *handing* would be derived systematically by combining the schemata for *hand* (namely, the PRED schema in (65)) with *ing*'s schemata (the PARTICIPLE specification) as the word is formed by suffixation.

15. A marking convention account of the defining/constraining distinction would have to provide an alternative lexical entry for each value that the vaguely specified feature could assume. A vague specification would thus be treated as an ambiguity, contrary to intuition.

16. In the more refined theory of lexical representation presented in chapters 1 and 3 of this volume, the relevant functions are those that appear in the function-assignment lists of lexical predicates. The two characterizations are essentially equivalent.

17. Unless, of course, the element is also the nonset value of another attribute. The point is that the element is inaccessible in its role as adjunct. An interesting consequence of this representation is that no cooccurrence restrictions between temporal adverbs and tense can be stated in the syntax, a conclusion justified independently by Smith 1978.

18. There is sometimes a preferred ordering of adjuncts and oblique objects. Grammatical descriptions might not be the proper account of these biases; they might result from independent factors operating in the psychological perception and production processes. See chapter 11 for further discussion.

19. The term *grammatical control* is sometimes used as a synonym for *functional control*. This kind of identification is distinct from *anaphoric control*, which links pronouns to their antecedents, and *constituent control*, which represents long-distance dependencies. Constituent control is discussed in section 4.7; for discussions of functional and anaphoric control, see chapters 5, 6, and 7.

20. In any event, the schemata in these alternatives violate the substantive restriction on functional control mentioned above. They also run counter to a second substantive restriction, the principle of *functional locality*. This principle states that for human languages, designators in lexical and grammatical schemata may specify no more than two function-applications. This limits the context over which functional properties may be explicitly stipulated. The recursive mechanisms of the c-structure grammar are required to propagate information across wider functional scopes. The locality principle is a functional analogue of the context-free nature of our c-structure grammars.

21. Grimshaw 1979a has argued that the sentential complement is restricted to be interrogative by the semantic type of the predicate WONDER. A separate functional specification of this restriction is therefore unnecessary.

22. Our controlled *e* is a base-generated analogue of the traces left by Chomsky's 1977b rule of *Wh* Movement. However, controlled *e*'s are involved only in the description of constituent control, whereas Chomsky's traces are also used to account for functional control phenomena.

Our controller and controllee metavariables also resemble the HOLD action and the virtual/retrieve arcs of the ATN formalism. Plausible processing models for both systems require similar computational resources to locate and identify the two ends of the control relationship. Thus, the experimental results showing that ATN resource demands predict human cognitive load (Wanner and Maratsos 1978, Kaplan 1975a) are also compatible with lexical-functional grammar. However, we discuss below certain aspects of our theory for which standard ATN notation has no equivalents: the appearance of controllees in the lexical entries of fully realized items, the root node specifications, and the bounding node conventions. Moreover, our theory does not have the characteristic left–right asymmetry of the ATN notation and thus applies equally well to languages like Basque, where constituent ordering is reversed.

23. Note as an aside that we have changed the Q-FOCUS identification schema from (135a) to (141) because the questioned element is no longer the ↓ f-structure of the fronted NP. The new schema places the interrogative semantic form in a canonical f-structure location that is independent of its degree of embedding. The complete fronted NP is also recorded in a canonical f-structure location, as the value of the function FOCUS. That NP is accessible as the FOCU of the question as well as through its clause-internal function OBJ, as indicated by the connecting line in (143). These separate access paths define the scope of different rules for interpreting anaphors. The FOCUS path in the f-structure for sentence (i) permits the ordinary pronoun *she* to be coreferential with *Sall* even though this is not permitted by its clause-internal object function, a shown by (ii):

(i) Which of the men that Sally dated did she hate ?
(ii) *She hated one of the men that Sally dated.
(iii) I wonder how proud of herself Bill thinks Sally is

The clause-internal function governs the interpretation of reflexive pronouns (iii) would otherwise be unacceptable because the reflexive is not a clause-mat of the antecedent *Sally*. The problem posed by the contrast between example (i) and (ii) was observed originally by Postal 1971. The solution sketched here i developed in greater detail by Zaenen 1980.

24. The lexical signature requirement and its formal implications are somewh reminiscent of Peters's 1973 Survivor property and Wasow's 1978b Subsistenc property, two restrictions that have been proposed to guarantee the recursiv ness of transformational grammars. Those conditions are imposed on the inp and output trees of a transformational cycle, whereas (147f) stipulates a prop erty that must hold of a single c-structure.

25. Chomsky 1977b proposes to derive such examples by restructuring rule that move the *of*-PP and *that*-complement outside of the *picture* and *claim* N before the *Wh* Movement rule applies. But such a reanalysis in all the releva cases cannot be justified, as Godard forthcoming shows for French.

26. Constituent control dependencies for relative pronouns also penetrate th genitive NP. This would follow automatically from the hypothesis that relati metavariables share the [+wh] subscript. The well-known distributional diffe ences between relative and interrogative items would be accounted for l additional features in the categorial subscripts for the relative and interrogati dependencies and more selective specifications on the linking schemata ass ciated with other bounding nodes.

27. Our use of linking schemata has some of the flavor of Chomsky's Su jacency Condition and COMP to COMP movement (Chomsky 1977b). W mentioned above that our specifications of bounding nodes differs from Cho sky's, but there are other significant differences in our approaches. For one, do not *move* constituents from place to place; we merely assert that a fur tional equivalence obtains. That equivalence enters into the f-description and

reflected in the ultimate f-structure, but it is never visible in the c-structure. Thus, we have a simple account of cases where *unmoved* constituents are subject to the bounded domination constraints, as in Chinese interrogatives (Huang 1980); in such cases, the theory of Chomsky 1977b fails to provide a uniform explanation.

28. We also leave open the possibility that a given controller has several domain roots. If several daughters of the controller node's mother are labeled with the controller's categorial superscript, then each such daughter becomes the root of a domain that must contain one corresponding controllee. This distributes the instantiation requirement to each of the domains independently. This suggests a plausible account for the across-the-board properties of coordinate structures, but more intensive investigation of coordination within the lexical-functional framework is needed before a definitive analysis can be given.

29. Chomsky 1977b has proposed an analysis of these sentences that does not involve a double dependency. He suggests an alternative phrase structure for examples of this type whereby the *on* PP belongs somewhere outside the *play* VP. Bach 1977 and Bresnan 1976c point out that this proposal has a number of empirical shortcomings.

30. The preposed item in relative clauses is also a TOPIC. Although the relative TOPIC might be functionally controlled when the clause is embedded next to the NP that it modifies, it must be linked anaphorically when the relative is extraposed.

31. The evaluation uses operators similar to Locate, Merge, and Include except that they return False whenever the corresponding solution operators would modify the f-structure.

32. Given the functional apparatus of our theory, we can demonstrate that the restrictions in (180) are necessary as well as sufficient for recursiveness. If nonbranching dominance cycles are allowed, there is a straightforward way of simulating the computation of an arbitrary Turing machine. The Turing machine tape is encoded in the f-structure, each level of which corresponds to one cell and has up to three attributes, CONTENTS (whose value is drawn from the TM's tape vocabulary), LEFTCELL (whose value is an encoding of the cell to the left), and RIGHTCELL. Each state of the TM is represented by a nonterminal category, and a transition from state $q_i$ to $q_j$ is represented by a rule rewriting $q_i$ as $q_j$. A single rule expands the starting category of the grammar to the initial state of the machine, and that rule has schemata that describe the TM's input tape. Starting at the top of the c-structure, each node in the nonbranching tree represents the next transition of the machine, and the f-structure at each node is the tape at that transition. The tape operations of a transition appear as schemata on the corresponding c-structure rule. They inspect the contents of the mother f-structure and produce an appropriate daughter f-structure. The lexical categories correspond to the final states of the machine, and the f-structure for a prelexical node is an encoding of the TM's output tape.

33. Certain other restrictions on metavariable correspondences will also provide this guarantee. For example, a nearly *crossed* restriction would also suffice, but it would entail more cumbersome models of processing. Formally what must be excluded is arbitrary degrees of nesting and crossing.

34. Bresnan, Kaplan, and Zaenen forthcoming discuss the formal consequences of the Dutch dependencies and provide a simple lexical-functional description of them.

35. The recursive specification in (190c) must be slightly complicated if f-structures are allowed to be cyclic, that is, to contain themselves as one of their attribute values, either directly or indirectly through some intervening f-structure levels. Structures of this kind would be induced by equations of the form $(f\,\alpha) = f$. If a Merge of two such structures is attempted, the recursive sequence might never reach a non-f-structure and terminate. However, any infinitely recursive sequence must repeat the merger of the same two f-structures within a finite number of steps. Merges after the first will have no effect, so the sequence can be truncated before attempting step (190c) for the second time. The Merge operator must simply keep a record of which pairs of f-structures it is still in the process of merging. The Locate operator is immune to this problem, since the number of its recursions is determined by the number of function-applications in the designator, which, being derived from the grammar or lexicon, is finite. While presenting no major formal difficulties, cyclic structures seem to be linguistically unmotivated.