NELSON CORREA

EMPTY CATEGORIES, CHAIN BINDING, AND PARSING

1. INTRODUCTION

In Government-Binding (GB) theory, the chief empirical effect of the principle Move-α is the definition of coindexing relations between nodes in phrase structure trees. This situation is explained by the extreme generality of the base component and several substantive constraints on transformations. The constraints on transformations, in particular Emonds' (1976) Structure Preserving Hypothesis, imply that the base may generate all S-structures of the core constructions of the language in question. The two conditions combine with a shift away from a grammatical model with both conditions on derivations, such as transformational rule ordering, and conditions on representations, such as the Subjacency condition on adjacent chain elements, to a grammatical model that favors representations.[1] It follows that the principal function of Move-α in the grammar is not reordering or otherwise rewriting of an input phrase marker for the derivation of S-structure, but rather the above noted establishment of coindexing relations between the nodes in the output representation. The relations in question are trace-antecedent relations. This trend in the role of Move-α in the grammar has been noted in the literature, especially by Koster (1978), Chomsky (1982), and Barss (1983). Koster proposes simplification of the class of grammars permitted in the Extended Standard Theory by eliminating the rule Move-α, reducing its empirical effect to a suitably extended theory of bound anaphora. Barss reformulates binding theory to remove inadequacies of the version in Chomsky (1981) in its account of clefts, relative clauses, questions, and topicalized constructions. Binding theory is generalized to constrain both argument and nonargument anaphoric relations, in particular those induced by application of the principle Move-α.

In this chapter we pursue the use of attribute grammars for the statement of computationally oriented instantiations of Government-Binding theory, exploiting the noted shift towards a representational

R. C. Berwick et al. (eds.),
Principle-Based Parsing: Computation and Psycholinguistics, 83–121.

model. We present an interpretive *Chain rule* for the identification of trace-antecedent relations. This rule is similar to that described in previous work (Correa, 1987), but differs from it in that the attribution statements in this case are oriented to bottom-up processing of the input string. We do not address in this chapter the problem of deriving a parsing program from the attribute grammar statement. This is a technical matter of great interest but no major linguistic relevance. It is partly addressed in the literature on programming languages and compiler construction; the reader is referred to Waite and Goos (1984), Kastens, Hutt, and Zimmermann (1982), and Watt and Madsen (1983). The new Chain rule is used in an English grammar written in the PLNLP language (Heidorn, 1972) and its operation may be observed in a bottom-up parallel parser generated automatically from the grammar by the PLNLP system.

This chapter is more explicit than Correa (1987) about the base generation and insertion of empty categories in phrase structure trees. The conditions assumed for classification of empty categories, as well as the correctness of the Chain rule proposed are assessed against the notion of Chain Binding proposed by Barss (1983). The chapter also addresses the recent trend in linguistic theory (especially within the GB framework) to specify natural language grammar in terms of extremely general generative rules, whose task is generation and annotation of syntactic structure, and whose output is subject to a variety of constraints or axioms that must be satisfied. As a case in point, we look at Barss' Chain Binding rule, an instance of this trend, and the Chain rule developed here. We suggest that Government-Binding theory, as currently proposed, is better viewed as an abstract specification of grammatical competence to which no psychological import is attached (insofar as processing is concerned). The translation of a competence grammar into computationally oriented mechanisms that satisfy the abstract specification is a difficult task, that has only recently begun to be addressed, and that should not be neglected since it explores another facet of the human language faculty.

The Chain rule may be evaluated in a direct way and is a crucial component of a performance model for Government-Binding theory. In contrast, transformations have proved to be the most significant obstacle for the formulation of analysis procedures for transformational grammar and the descendent GB theory . While transformations are useful grammatical devices, amenable to algorithmic implementation when viewed

as generative devices—*i.e.*, in the sense of tree transduction—there is no general directed recognition procedure for the languages defined by transformational grammars. Thus, we find in Petrick (1965) and more recently Sharp (1985) recourse to a covering surface structure grammar and inverse transformations to implement a generate-and-test analysis procedure.

In this chapter we assume basic knowledge of Government-Binding theory and motivation for the grammatical principles assumed in it (Chomsky, 1981; van Riemsdijk and Williams, 1986). For readers interested in parsing, we also assume knowledge of compiler construction techniques from a given attribute grammar language definition (Waite and Goos, 1984).

The chapter is organized into eight sections as follows. In section 2 we sketch our assumptions about phrase structure and the manner in which empty categories arise. In section 3 we define attribute grammars and illustrate their use in the definition of core grammatical processes in Government-Binding theory, namely government and Case marking. Section 4 discusses the distribution of empty categories according to functional type and casts within the framework of attribute grammar alternatives about the determination of empty categories, according to whether their syntactic features are intrinsically, functionally, or freely determined (Lasnik and Uriagereka, 1988). Section 5 presents the bottom-up version of the Chain rule and illustrates its operation with an example. In section 6 we present Barss' (1983) Chain Binding algorithm and compare it to the Chain rule. We claim that the two rules are empirically equivalent, so the relation between them is that of program refinement. In section 7 we give a simple demonstration about the exponential complexity of the generate-and-test algorithm assumed in Barss' Chain Binding, compared with the linear complexity of the more directed Chain rule. This leads in section 8 to a distinction between grammars as definitions of a language, versus grammars as concrete mechanisms for the generation and analysis of strings in a language. This point is illustrated with the aid of an artificial language.

2. PHRASE STRUCTURE AND EMPTY CATEGORIES

We assume a base component that satisfies the principles of the $\overline{X}$ theory, a restricted form of context-free grammar (Chomsky, 1970). Pullum (1985) surveys the variants of $\overline{X}$ grammar. The principal notion that $\overline{X}$

theory captures is that of *head of a phrase*; in particular it requires that every syntactic category be a *projection* of a head. Adopting current conventions about $\overline{X}$ grammar, we assume a two-level system, in which zero-level categories are denoted by X, first-level projections by $\overline{X}$, and second-level or maximal projections by XP. Complements of a head are generated as sisters of the head, while specifiers and adjuncts are sisters of the first-level projection. The relative order of complements, specifiers, and adjuncts with respect to the head is a parameter of universal grammar. The base component in a GB grammar is thus defined by the maximally concise $\overline{X}$ production schema (1), assuming the parameter settings for English.

(1) (a) $XP \rightarrow YP\ \overline{X}$

(b) $\overline{X} \rightarrow X\ YP$

The zero-level categories in the grammar are comprised of the *lexical* categories (2a), whose extension is defined by the entries in the lexicon, and by the *nonlexical* categories (2b), that do not have lexically defined extensions, but rather are expanded into certain specified lexical formatives or the empty string. The schema (1) specify that in English specifiers precede their heads, while complements follow them. The schema are extended with (3) to provide an arbitrary number of positions for adjuncts, typically following the head (in English).

(2) (a) N(oun), A(djective), V(erb), P(reposition), Adv(erb), Q(uantifier), Art(icle), M(odal)

(b) I(nflection), C(omplement)

(3) $\overline{X} \rightarrow \overline{X}\ YP$

Attribute grammars define semantic and context-sensitive properties of a language in a *syntax-directed* manner; that is, by making direct reference to phrase structure rules for the statement of attribution rules and conditions. Accordingly, the base component that we assume is the explicit set of context-free productions defined by the schema (1) and (3). The $\overline{X}$ schema are overly general, though. They permit any maximal projection YP to appear as a complement, specifier or adjunct of a given head X. Typically, a head X allows only certain categories to appear in these positions. Subcategorization or thematic properties of the head

may restrict the categories that appear as complements, but other lexical properties of a head do not in general determine the category of the specifiers or adjuncts that it allows. Here we assume that the $\overline{X}$ schema is supplemented with other devices (*e.g.*, a table) indicating the categories allowed at each position for each head X; Jackendoff (1977) addresses the issue at length.

The structure of projections of the nonlexical categories I and C is crucial in the statement of the Chain rule in section 5. The category I introduces tense, modal, and inflectional elements of a sentence and is the head of the sentence symbol IP.[2]. The category C provides a position for the clause complementizer in pre-IP position, and projects to the category CP for clause. Adopting current assumptions about clause structure (Chomsky, 1986), we identify the position for *Wh*-movement with the C-specifier position, rather than the complementizer position itself.[3] The specific phrase structure productions for projections of I and C appear in (4) and (5).

(4) (a) IP $\rightarrow$ NP $\bar{\text{I}}$

(b) $\bar{\text{I}}\rightarrow$ I VP

(5) (a) CP $\rightarrow$ (NP| PP| CP) $\overline{\text{C}}$

(b) $\overline{\text{C}}\rightarrow$ C IP

The nonlexical expansions of I and C are given by the productions (6). The first righthand side alternative in (6a) is redundant, but we show it explicitly for clarity.

(6) (a) I $\rightarrow \epsilon$ | (M)(*have*)(*to*)(*be1*)(*be2*)

(b) C $\rightarrow \epsilon$ | *for* | *that* | *whether*

As implied by the $\overline{X}$ schema (1) and (3), most base productions are binary branching. Note that (6a) could easily be stated in binary form. We ignore certain details regarding the schema that are required to permit several constituents to be attached as complements, specifiers, or adjuncts of the same head. To illustrate, (1b) permits attachment of only a single constituent as complement of the head X. But for many languages most analyses assume that more than one such constituent may be attached. The 'abstract syntax' thus contains the nonbinary branching scheme (7) in place of (1b). The 'concrete syntax' may be

captured succinctly by a binary-branching scheme with recursion on a category symbol, for example as in (8). Recursion is via the category X of the head; constituents are attached as sisters of this symbol, one at a time, until the subcategorization/thematic requirements of the head are satisfied. Binary branching rules have certain advantages over nonbinary ones, particularly describing free constituent order, optional constituents, and occurrence of nonarguments, such as adjuncts and parentheticals, between the complements of a head (Jensen, 1987). More traditional accounts of these phenomena assume unordered righthand sides of productions, unit productions, or construction-specific transformations, such as NP-shift.

(7) $\overline{\text{X}} \rightarrow \text{X YP}_1 \ldots \text{YP}_n$

(8) (a) $\text{X} \rightarrow \text{X YP}$
(b) $\overline{\text{X}} \rightarrow \text{X}$

In addition to the productions implied by the schema (1) and (3), the base contains productions that rewrite a symbol X into the empty string ϵ, as in (9). The schema (9) has instances for X equal to C, I, V, NP, PP, CP, and possibly other categories. A phrase marker $[_{\text{X}}\ \epsilon]$ derived by application of one such rule is called an *empty category* of category X.

(9) $\text{X} \rightarrow \epsilon$

An empty category generated by (9) is said to be *base generated*. In the model of transformational grammar, empty categories also arise due to the operation of transformations. Application of Move-α or Quantifier Raising (QR) leaves an empty category (a *trace*) at the extraction site and crucially induces a syntactic relation between the moved constituent and the trace. By the trace convention, the trace-antecedent relation is represented by coindexation of the two elements. The moved constituent together with the traces coindexed with it are said to form a *syntactic chain*. As noted in the introduction to this chapter, it is not crucial to the notion of trace-antecedent relations that they are transformationally defined. Instead, we pursue here the idea that they are anaphoric relations, established by an interpretive rule.

Base rules like (9) are also needed in a transformational model, assuming a strict version of Emonds' (1976) Structure Preserving Hypothesis. This constraint requires a target position $[_{\text{X}}\ \epsilon]$ in the input phrase

marker for movement of another phrase marker [$_X$ α]. Under this interpretation, Move-α is structure preserving, but Quantifier Raising is not.

3. ATTRIBUTE GRAMMARS

Attribute grammars are an extension of context-free grammars, originally developed for the formal specification of the semantics of programming languages and their compilers. Two closely related formalisms, developed with the same purpose in mind and at about the same time, are the Affix grammars of Koster (1971) and the Two-level grammars of van Wijngaarden (1969). Although the generative power of these formalisms is equivalent to that of unrestricted rewriting systems, they are of significant linguistic interest, in the sense used by Chomsky (1959), since they permit suitable structural descriptions to be associated with the strings they generate.[4]

The first formal characterization of attribute grammars is due to Knuth (1968), although the main ideas can be traced back to the syntax-directed compiler of Irons (1961). Attribute grammars appear related to the unification-based formalisms discussed by Shieber (1986) in one important respect: namely, the basic 'informational elements' in unification formalisms are attribute-value structures of the sort that appear in attribute grammars. Attribute-value structures are in turn an elaboration of the feature matrices used for phonological description. The two formalisms differ significantly, however, in the kinds of operations allowed to describe attribute values and in their separation of phrase structure from attribution structure. The characterization of attribute grammars in the present section is adapted from Waite and Goos (1984).

3.1. *Formal Definition of Attribute Grammars*

An *attribute grammar* is defined using a context-free grammar $G = (N, T, P, Z)$, where N and T are nonterminal and terminal vocabularies, respectively, P is a finite set of productions, and Z the start symbol. The grammar defines for each symbol $X \in N \cup T$ a set $A(X)$ of *attributes* or syntactic features, and a *type* or domain $dom(a)$ of possible values for each attribute $a \in A(X)$.

Each attribute represents a specific, possibly context-sensitive prop-

erty of symbol X. For each occurrence $X.a$ of the attribute in a derivation tree containing X, its value must be uniquely defined.[5] An *attribute occurrence* is a single-assignment variable.

Attribute values are defined by *attribution rules* of the form $X_i.a \leftarrow f(X_j.b, \ldots, X_k.c)$, associated with each production $p = X_0 \rightarrow X_1 \ldots X_n$ in the grammar, $0 \leq i, j, k \leq n$. The intent is that f is an effectively computable applicative expression whose value depends only on the values of attribute occurrences associated with symbols in p. In particular, f cannot perform any kind of tree traversals to compute its value.[6] When p applies in a derivation, the attribution rule defines the value of attribute occurrence $X_i.a$ in terms of the occurrences $X_j.b$, ..., $X_k.c$, associated with other symbols in p. We let $R(p)$ denote the packet of attribution rules associated with p.

In addition to attribution rules, the grammar may define *attribute conditions* of the form $B(X_i.a, \ldots, X_j.b)$, for $0 \leq i, j \leq n$, on the attributes of symbols occurring in the production p. These conditions are truth-valued functions that must be satisfied in derivation trees requiring application of the production, and thus contribute to the notion of *grammaticality* in the language generated by the attribute grammar. Otherwise plausible constructions may be ruled out by the violation of one or more attribute conditions. We let $B(p)$ denote the packet of attribute conditions associated with p.
The above remarks are summarized in definition (10):

(10) An *attribute grammar* is a four-tuple $AG = (G, A, R, B)$, where

- (a) $G = (N, T, P, Z)$ is a context-free grammar,
- (b) $A = \bigcup_{X \in V} A(X)$ is a finite set of *attributes*,
- (c) $R = \bigcup_{p \in P} R(p)$ is a finite set of *attribution rules* of the form $X.a \leftarrow f(Y.b, \ldots, Z.c)$, and
- (d) $B = \bigcup_{p \in P} B(p)$ is a finite set of *attribute conditions* of the form $B(X.a, \ldots, Y.b)$.

The underlying context-free grammar G assigns a derivation tree to each sentence in $L(G)$. Each node X in the tree is annotated with the set $A(X)$ of attributes associated with X. If the grammar is well-defined, in the sense given below, then it is possible to evaluate each attribute occurrence on the tree. A tree generated by G is *correctly attributed* if upon attribute evaluation, all attribute conditions yield *true*. We define

the *language* generated by the attribute grammar, $L(AG)$, as the set of sentences in $L(G)$ that have at least one correctly attributed tree.[7]

For each production p in G, we let the set $AF(p) = \{X_i.a : X_i.a \leftarrow f(\ldots) \in R(p)\}$ be the set of *defining occurrences* of attributes associated with symbols in p. The attributes associated with a symbol X are classified according to the manner in which their values depend on attributes in the neighboring nodes. We say that attribute occurrence $X.a$ is *synthesized* if its value depends only on values of attributes on daughters of X, and possibly also other attributes of X. The attribute occurrence is *inherited* if its value depends on the values of attributes associated with the parent or sister nodes of X. Thus, synthesized attribute values result from consideration of the subtree dominated by the symbol X, while inherited attribute values depend on the local syntactic environment (parent and sisters) of X. This classification of attributes in a grammar is stated precisely in (11).

(11) Let $AG = (G, A, R, B)$ be an attribute grammar. An attribute $a \in A$ is *synthesized* if there exists a production $p = X \rightarrow \gamma$ such that $X.a \in AF(p)$. The attribute is *inherited* if there is a production $q = Y \rightarrow \alpha X \beta$ such that $X.a \in AF(q)$.

We let $AS(X)$ and $AI(X)$ denote the sets of synthesized and inherited attributes of X, as in (12).

(12) $AS(X) = \{X.a :$ for some $p = X \rightarrow \gamma \in P, X.a \in AF(p)\}$
$AI(X) = \{X.a :$ for some $p = Y \rightarrow \alpha X \beta \in P, X.a \in AF(p)\}$

We admit certain attributes, such as the lexical features of a lexical item, whose values are determined by the symbol with which they are associated. We call these attributes *intrinsic* or inherent, since their values are defined in the lexicon. This class, however, may be reduced to inherited and synthesized attributes without loss of generality.[8]

3.2. *Well-formedness of Attribute Grammars*

Nothing thus far prevents writing attribute grammars that yield improperly attributed trees. This happens when some occurrence of an attribute in a tree has its value defined multiple times (in possibly multiple ways), or does not have a value defined at all.

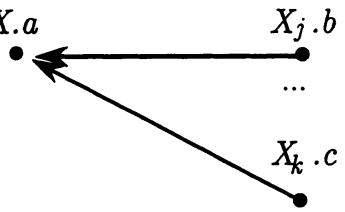
(13) An attribute grammar $AG = (G, A, R, B)$ is *consistent* if for each attribute occurrence $X.a$ in a derivation tree generated by

G, at most one attribution rule is applicable to compute its value. The grammar is *complete* if at least one such attribution rule is applicable.

To illustrate the notion of consistency, the *Case* value assigned to a noun phrase (NP) must be uniquely defined. If several mechanisms exist for Case assignment to a given NP position, at most one may apply. Thus, an embedded subject position may be Case-marked by *tense*, a *for* complementizer, or an exceptional Case marking verb. The interaction of these mechanisms must be such that at most one applies. If an attribute grammar is consistent, then for each symbol X in its vocabulary the sets $AS(X)$ and $AI(X)$ are disjoint: $AS(X) \cap AI(X) = \emptyset$. If the grammar is complete, then $A(X) = AS(X) \cup AI(X)$, for each X; we obtain a partition of the set of attributes associated with each symbol X into synthesized and inherited attributes (Waite and Goos, 1984).

Well-formedness of an attribute grammar also depends on the nature of the data dependencies between attributes. Let $X.a$ be an attribute occurrence whose value is defined by the attribution rule $X.a \leftarrow f(X_j.b, \ldots, X_k.c)$. In order to compute the value of $X.a$, we must know the values of attribute occurrences $X_j.b, \ldots, X_k.c$ upon which $X.a$ depends.[9] In this case we say that $X.a$ *directly depends* on $X_j.b, \ldots, X_k.c$. This situation is shown graphically in the directed graph (14), where the node set contains the attribute occurrences involved and the edge set contains the direct attribute dependencies.

(14)



In general, given a structure tree τ with node set K and collection D of direct dependency relations between attributes, a *dependency graph* $DT(\tau) = \langle \bigcup_{X \in K} A(X), D \rangle$ is defined that records all the dependency relations arising from the attribution associated with the productions applied to derive the tree. We are now in a position to define a *well-defined* attribute grammar, as in (15).

(15) An attribute grammar is *well-defined* if and only if it is consistent, complete, and the dependency graph $DT(\tau)$ is acyclic for

each derivation tree τ corresponding to a sentence of $L(G)$.[10]

3.3. *Attributed Definition of Grammatical Processes*

Government-Binding is a modular theory in which phrase structure rules play only a limited, though basic role in the definition of a language. Following an argument by Stowell (1981), most current work assumes that the phrase structure component is eliminated—that is, reduced to the $\overline{\text{X}}$ schema (1), (3), and (9) (Lasnik and Uriagereka, 1988). Attribute grammars can be used to capture core grammatical processes, such as government and Case marking. Each process defines formally part of the overall attribution on the trees generated by the grammar.

3.3.1. *Government*

We consider government first. The core notion is the relation between the head of a phrase and each of its complements and specifiers, as defined in (16). The relation *m-command* (maximal command) used in (16ii) is defined in (17).

(16) α *governs* β if

- (i) α is a zero-level category; if α is I(nflection), then AGR has a non-*nil* value.
- (ii) α m-commands β.
- (iii) Each maximal projection dominating β also dominates α.

(17) α *m-commands* β if

- (i) Neither α nor β dominates one another.
- (ii) The first maximal projection dominating α also dominates β.

If the head in question is of category I, then the agreement element AGR must be present. Thus, the subject of a finite sentence is governed by the inflectional element, while the subject of an infinitival sentence is ungoverned (by that element). Condition (16i) must be carefully formulated. The agreement attribute AGR is in the attribution set of each category that exhibits agreement phenomena, as shown in (18a). The domain of AGR is (18b); it includes the value *nil*, which is taken by AGR when the category in question does not have 'any' agreement features.

(18) (a) AGR is in $A(X)$, for X=N, $\overline{\text{N}}$, NP, I, $\bar{\text{I}}$, V, $\overline{\text{V}}$, VP.

(b) $dom(AGR) = \{nil\} \bigcup \{ \langle P, G, N \rangle : P = 1,2,3, G = masc,$ *fem*, *neuter*, and N= *plural*, *sing*$\}$

By standard assumptions, only maximal projections may appear as complements, adjuncts, or specifiers of a head. It follows from condition (16iii) that only maximal projections may be governed. Current usage of the notion of government, however, assumes that if α governs XP, then α also governs the head X and the intermediate projection $\overline{\text{X}}$. The core notion (16) of government is extended as in (19).

(19) If α governs β, then α also governs the head of β and its intermediate projections.

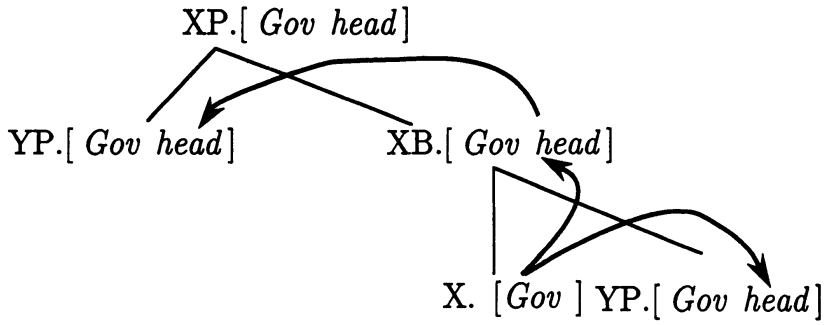
By this extension, a node of any category may be governed. We assume an attribute *Gov* that is associated with all categories in the grammar. According to the definitions, a node may be governed in the core sense of (16) and (19) by only one other node. An appropriate domain for *Gov* is pointers to the annotated tree nodes defined by the grammar. The attribute evaluates to a pointer to the governor of the node that bears the attribute occurrence. If there is no governor for the node, then *Gov* evaluates to the special value *nil*. The core process of government is captured by the attribution in (20a–b), with direct reference to the $\overline{\text{X}}$ schema (1). If X is a symbol in a grammar rule, we use the notation $\uparrow$X to denote a pointer to the record structure of the node corresponding to symbol X.[11] The attribute *head* in the attribution is associated with all nonzero level categories and its domain is pointers to record structures labeled by zero-level categories—*i.e.*, heads of phrases. (20c) shows the dependencies induced by government; similar dependency graphs may be drawn for attribution statements in the remainder of the chapter.

(20) (a) $\overline{\text{X}} \rightarrow$ X YP
attribution:
$\overline{\text{X}}.head \leftarrow \uparrow$X
YP.$Gov \leftarrow \uparrow$X
X.$Gov \leftarrow \overline{\text{X}}.Gov$

(b) XP$\rightarrow$YP $\overline{\text{X}}$
attribution:
XP.$head \leftarrow \overline{\text{X}}.head$

YP.*Gov*←$\overline{\text{X}}$.*head*
$\overline{\text{X}}$.*Gov*←XP.*Gov*

(c)



The category variable X in (20) ranges over lexical categories. For X equal to I(nflection), the attribution may not be obtained from the same schema. Instead, the attribution should be sensitive to the presence of the *AGR* attribute on the I node, as in (21). The agreement feature is propagated upwards from the I to the $\overline{\text{I}}$ node. If I.*AGR*≠*nil*, then I governs the NP subject. If I.*AGR*=*nil*, however, condition (16i) of government does not obtain, and I does not govern. The Subject position may still be governed exceptionally, by a *for* clause complementizer, if present, or by the governor of the clause, if it is an exceptional Case marking (*ecm*) verb such as *want* or *believe*. The governor of the IP node is given by either of these two possibilities, or is *nil* otherwise. The attribution in (22) implements this assumption. The values of the *Gov* and *ecm* attributes of the CP node in (22b) are, in turn, determined by the context in which the node appears.

(21) (a) $\overline{\text{I}}$→I VP
attribution:
$\overline{\text{I}}$.*head*←↑I
$\overline{\text{I}}$.*AGR*←I.*AGR*

(b) IP→NP $\overline{\text{I}}$
attribution:
NP.*Gov*← if $\overline{\text{I}}$.*AGR*≠*nil* then $\overline{\text{I}}$.*head* else IP.*Gov*
IP.*head*←$\overline{\text{I}}$.*head*

(22) (a) $\overline{\text{C}}$→C IP
attribution:

IP.*Gov*←if C.*forcomp* then ↑C
else if $\overline{\text{C}}$.*ecm* then $\overline{\text{C}}$.*Gov*
else *nil*
$\overline{\text{C}}$.*head*←↑C

(b) CP→(XP) $\overline{\text{C}}$
attribution:
$\overline{\text{C}}$.*Gov*←CP.*Gov*
$\overline{\text{C}}$.*ecm*←CP.*ecm*
CP.*head*←$\overline{\text{C}}$.*head*

We have ignored proper government in this discussion, in particular government by a local antecedent. Given a suitable definition of this notion, the manner in which it by may expressed in an attributed definition seems clear, although it certainly may raise some questions. For example, subject position may be governed by a tensed inflection node, but properly governed only by an element in C-specifier position. It is necessary to identify the precise mechanism by which the relevant governor is chosen.

3.3.2. *Case Marking*

Grammatical Case is assigned to a noun phrase by a Case assigner structurally, under government, or inherently, in double-object constructions (Chomsky, 1981). The instances of structural Case assignment are: (i) assignment of nominative Case to an NP subject, if governed by a tensed I node (if the node is not tensed, subject position may still be Case-marked exceptionally, as detailed below); (ii) objective Case assignment by a verb; (iii) oblique Case assignment by a preposition; and (iv) genitive Case marking of an NP in N-specifier position.

Structural Case assignment is captured by the attributed definitions (23)–(24). The attribute *Case* is associated with NP, the categories of potential Case assigners, such as V and P, and several other categories that may intervene in the process, including CP and IP. The domain of the attribute is (25), which includes the special value *nil*. A node X is said to have 'no Case' if X.*Case* has the value *nil*.

(23) (a) IP→NP $\overline{\text{I}}$
attribution:
NP.*Case* ← if $\overline{\text{I}}$.*tensed* then *Nom* else IP.*Case*

(b) $\overline{X} \rightarrow X\ YP$
attribution:
$YP.Case \leftarrow X.Case$

(c) $NP \rightarrow NP\ \overline{N}$
attribution:
$NP2.Case \leftarrow Gen$

(24) (a) $\overline{C} \rightarrow C\ IP$
attribution:
$IP.Case \leftarrow$ if $C.forcomp$ then $Oblq$
else if $\overline{C}.ecm$ then $\overline{C}.Case$ else nil

(b) $CP \rightarrow (XP)\ \overline{C}$
attribution:
$\overline{C}.Case \leftarrow CP.Case$

(25) $dom(Case) = Nom, Acc, Dat, Oblq, \ldots, nil$

The attribution (23a) defines the first instance of structural Case assignment. It interacts with (24), which defines the mechanism of exceptional Case marking. Notice that CP deletion is not postulated, but rather that the categories CP, $\overline{C}$, and IP bear the *Case* attribute, and are instruments in propagating a Case value from an exceptional Case assigner to the exceptionally Case-marked NP. This formalization is similar to that of van Riemsdijk and Williams (1986), who assume that a maximal projection may be lexically marked 'transparent' to government and Case marking. (23b–c) cover the other instances of structural Case assignment. Inherent Case assignment, not covered by (23), may be added similarly, although the actual motivation for inherent Case marking is controversial.

The Case filter, a major constraint on NP distribution, is simply expressed by attribute condition (26). It is associated with all productions that expand NP.

(26) Case filter:
if $\neg NP.empty$ then $NP.Case \neq nil$

4. DETERMINATION OF EMPTY CATEGORIES

In the interpretive model of grammar adopted here, all empty categories arise by base generation. The categories generated, however, exhibit dif-

ferent behavior, as they appear in different environments. Government-Binding theory distinguishes four types of empty category (of syntactic type NP): *Wh*-trace, NP-trace, PRO, and small *pro*. In a transformational account, the first two are instances of trace, and arise due to the operation of Move-α. The last two categories are base generated and behave like empty pronominals. Small *pro* does not occur in English, however.

The functional type of an empty category may be encoded with two attributes *anaphoric* and *pronominal*, as in (27), like the way overt nominals are partitioned. *Wh*-trace is nonanaphoric and nonpronominal (a referential expression); NP-trace is anaphoric and nonpronominal (a pure anaphor); and PRO is both anaphoric and pronominal (with no counterpart in overt nominals).[12]

(27)

	Wh-trace	NP-trace	pro	PRO
anaphoric	−	+	−	+
pronominal	−	−	+	+

An empty category's type may be functionally determined by the category's environment. Chomsky (1982) refers to the argument status of the position in which the category appears and to properties of its antecedent, if it has one. Thus *Wh*-trace is in argument position (A-position) and bound by an element in nonargument position ($\overline{\text{A}}$ position, $\overline{\text{A}}$-bound). NP-trace appears in argument position and is A-bound by an element in a position at which no thematic role is assigned ($\overline{\theta}$-bound). PRO is in argument position, and is optionally A-bound by an element with independent thematic role (θ-bound).

A crucial property of Chomsky's functional determination of empty categories is that it requires reference to the antecedent of the category, if it has one, before its type may be determined. In Correa (1987) a different set of contextual conditions is used to determine the functional type of an empty category without reference to its antecedent—that is, before chains are hypothesized. These conditions are the argument status of the position, government, and Case. Hence, a trace is (properly) governed, while PRO is not. Two instances of *Wh*-trace are recognized: variables and intermediate traces in C-specifier position.[13] Variables are in argument position and Case marked, while NP-trace is also in argument position, but not Case marked. A *Wh*-trace in C-specifier position

is, by definition, in nonargument position. This yields the method (28) for functional classification of empty categories, according to the new conditions.

(28)

	A-position	Government	Case
Wh-trace (variable)	+	+	+
Wh-trace (C-spec)	−	+	−
NP-trace	+	+	−
PRO	+	−	−

The fact that (28) does not refer to antecedents to determine functional type is extremely useful for the formulation of interpretive analysis procedures for the grammar. It means that we may determine the type of a category without hypothesizing in advance the chain to which the category belongs. This avoids an inefficient generate-and-test approach for the interpretive computation of the chains present in a given surface string. Instead, once the relevant phrase structure is identified, it is sufficient to look at the argument status of the position, and whether it is governed and Case-marked—all three of which are conditions established independently of chain formation. The analysis (28) is a crucial element of the interpretive Chain rule in Correa (1987), and of the bottom-up version of the rule developed in the next section.

Before leaving this section, however, it is of interest to discuss alternatives for the determination of empty categories. An interpretation of the approach (28), and also that in Chomsky (1982), is that empty categories are base generated (or transformationally inserted) as categories of type NP, without instantiation of the values of the features that determine their functional type. That is, at generation time there is only one type of empty category: empty NP with unspecified functional type. Scheme (9) generates empty categories and may be augmented by the attribution (29), so that it defines the values of the attributes *anaphoric* and *pronominal*. The functional type of a category is determined when it is generated in a particular context. The attributes *anaphoric* and *pronominal* are synthesized by NP, but their values may not be determined until the values of the attributes *Apos*, *Gov*, and *Case* are instantiated. The latter are instantiated only when the NP is inserted in a particular context.[14]

(29) NP$\rightarrow \epsilon$

attribution:
NP.*anaphoric*←if NP.*Apos*='–' then '–'
else if NP.*Gov*=*nil* then '+'
else if NP.*Case*=*nil* then '+' else '–'
NP.*pronominal*← if NP.*Gov*=*nil* then '+' else '–'

A second interpretation of (28) is as a set of conditions on the contexts in which an empty category of a given type may be inserted. Under this approach, the type of an empty category is determined when it is generated. The attribution in (30) implements this approach. The attribution rules fix functional type and may be evaluated as soon as the production is applied, while the attribute conditions constrain the contexts in which the category may appear, and may be evaluated only once the context is established. This new attribution parallels the 'intrinsic' determination of empty categories assumed in Chomsky (1981) and discussed by Lasnik and Uriagereka (1988), and is completely equivalent to the attribution (29) in terms of the annotated trees it defines.[15]

(30) NP→ ϵ
attribution:
NP.*anaphoric*←x
NP.*pronominal*←y, for x, y='+', '–'
condition:
NP.*anaphoric*=if NP.*A-pos*='–' then '–'
else if NP.*Gov*=*nil* then '+'
else if NP.*Case*=*nil* then '+' else '–'
NP.*pronominal*=if NP.*Gov*=*nil* then '+' else '–'

In a transformational model, intrinsic determination of empty categories is problematic in the following case. The functional type of a category left by Move-α is presumably determined by lexical features of the moved phrase, since the structural description of Move-α does not indicate whether movement is to an argument or nonargument position. Otherwise two distinct movement transformations would need be posited. But then successive movement of the same phrase may leave two traces of different functional type, as in (31), which is NP-movement followed by *Wh*-movement. Thus, intrinsic determination is not possible in a transformational definition of chains, without unduly complicating the transformational component. Functional determination, as given by (28) or Chomsky (1982), must be assumed.

(31) Who$_i$ was $[_{IP}\ [\epsilon]_i$ beaten $[\epsilon]_i\ ]$

A third alternative for the determination of empty categories is free-determination. As described in Lasnik and Uriagereka (1988), this is similar to functional determination in that there is only one 'type' of empty category at generation time, namely an empty NP with functional type unspecified. However, rather than having an algorithm that functionally determines the type of a category at insertion time, there is a simpler algorithm that 'freely' assigns types to categories (at some level of representation), while "independent principles will rule out the bad constructions" (Lasnik and Uriagereka, 1988, p. 66). In this respect, then, free determination is actually closer to intrinsic determination. The independent principles needed must include or subsume the contextual conditions (28) and those of Chomsky (1982). From a computational viewpoint there appears to be no important difference between free determination and the previous two approaches.

5. INTERPRETIVE CHAIN RULE

The base component generates all possible S-structures. Traces, as well as antecedents, are base generated in their surface positions. We refer to an S-structure in which trace-antecedent relations are not identified as a *surface structure*. Given a structure in this sense, the objective of the Chain rule is to identify interpretively the trace-antecedent relations that may be present in that structure. We assume an attribute *Chain*, associated with NP, whose domain is pointers to annotated tree nodes. The Chain rule evaluates the *Chain* attributes of the NPs in the given surface structure. Upon evaluation, if the attribute is associated with a node X, it points to the next element in the chain to which X belongs, or is *nil* if X is the last element of the chain. This is illustrated with a simple example in (32). Application of the Chain rule yields an S-structure.[16]

(32) $[_{NP}$Who$]_{.Chain}$ does John think$[_{CP}[_{NP}\epsilon\ ]_{.Chain}$ Mary likes$[_{NP}\epsilon\ ]_{.nil}\ ]$

The functional type of an empty category may be determined in advance of the application of the Chain rule, by the method of last section. We note that the method uniquely determines the type of the category, but that nonstructural ambiguity may be present regarding government and Case marking. For example, in (33) there is ambiguity

regarding the type of the empty category in embedded subject position, depending on whether the matrix verb *want* exceptionally governs and Case marks the position. In (33a), *Wh*-trace and PRO are the only two possibilities. NP-trace is not feasible, since if the position is governed it is also Case-marked by *want*. The ambiguity between *Wh*-trace and PRO disappears once the Chain rule (or alternatively, well-formedness conditions on chains) applies. *Wh*-trace is not possible since it would lack an antecedent; PRO is the only possibility. In the closely related (33b), the situation is reversed. PRO is not possible, since this would leave a vacuous operator *who*; hence *Wh*-trace is the only alternative. In (33c), similar considerations indicate that NP-trace and PRO are both plausible, but only NP-trace satisfies all considerations.

(33) (a) John wants $[_{CP}\ [\epsilon]$ to do the dishes]

(b) Who$_i$ does John want $[_{CP}\ [\epsilon]_i$ to do the dishes]

(c) John$_i$ is wanted $[_{CP}\ [\epsilon]_i$ to do the dishes]

We distinguish three kinds of chains: elementary, argument, and nonargument. Elementary chains are a special instance of argument chains and contain a single NP element. If the NP is lexical, it must be in a position which is Case-marked and θ-marked. An expletive element is exceptional, since it requires no θ marking. If the NP is empty, it must be PRO and θ-marked. In general, argument chains consist of elements in argument position, and are characterized by the fact that their last element is a trace in a θ-marked position.[17] Since the element must be NP-trace, the position is Caseless. The θ-criterion and the Projection Principle imply that any other chain element, including the head, is in a non-θ-marked position. Thematic theory determines that subject position (of IP) is the only possibility for the other chain elements. If the chain is headed by a lexical NP, Case theory requires that exactly one position be Case-marked; Chomsky (1986) assumes that position is always the head.

Nonargument chains are headed by an operator in C-specifier position or by a phrase in Topic position. Every other element of the chain, except the last, is a trace in C-specifier (nonargument) position. The last element is a trace in Case-marked argument position, and is either θ-marked, or the head of an argument chain. In section 6 we will discuss an additional locality condition on chain bindings.

In addition to the attribute *Chain* associated with NP, and corresponding to the two types of nonelementary chain, the Chain rule assumes a two pointer-valued attributes *A-Chain* and *AB-Chain* (corresponding to the linguistic notions of an A-chain and an $\overline{\text{A}}$-chain). These attributes are associated with the nodes CP, $\overline{\text{C}}$, IP, $\overline{\text{I}}$, VP, $\overline{\text{V}}$, PP, and $\overline{\text{P}}$, all of which may be in the c-command domain of NP, for some tree generated by the base.[18] It is sufficient to consider only these nodes for the propagation of chain links, since a computationally important property of movement, which follows in part from the Binding theory of Chomsky (1981), is that movement is always to a c-commanding position.

The Chain rule defines how the attributes *Chain*, *A-Chain*, and *AB-Chain* are evaluated locally in terms of each other and other properties of an S-structure tree. More abstractly, the Chain rule defines local constraints on the possible values of these attributes. The domain of evaluation is always limited to an elementary tree defined by a production in the grammar. From this observation, it is easy to see that the account of chains given by the Chain rule always reduces to simple relations between attribute occurrences in elementary trees. Informally, the steps of the Chain rule are as in (34). Note, however, that attributed definitions do not imply a particular ordering of attribute evaluation steps; (34) is a sketch of how evaluation might proceed.

(34) Chain rule (sketch)
For each NP in the input structure,

1. Determine the functional type of the NP.

2. Assign the NP to a chain:
Evaluate the *Chain* attribute of the NP, according to its functional type and other properties of the position it occupies, including θ-role.

3. Start, propagate, or terminate a chain:
Evaluate the *AB-Chain* and *A-Chain* attributes on the parent of the NP. The evaluation depends on properties of the NP and its position.

The attribution that defines the Chain rule is stated in (35), (37), and (38) below. (35) shows the schema for the productions that introduce complements of a head X. The predicates *Wh-trace* and *NP-trace* in (35a) evaluate to true, according to the type of the NP. If X is a

symbol with several occurrences in a production, the notation X_i, for $i \geq 0$, denotes the i-th occurrence of X on the righthand side, or the symbol X on the lefthand side, if $i = 0$. We employ PLNLP notation in the attribute conditions, by which a Boolean test like $\neg X.a$ on attribute $X.a$ succeeds if $X.a$=*nil*. An important property of (35) and subsequent attribution statements is that attribute evaluation is local; that is, limited to attribute occurrences on the elementary tree defined by the production in question.

(35) (a) X→ X NP
attribution:
NP.*Chain*←*nil*
X_0.*AB-Chain*←if *Wh*-trace(NP) then ↑NP
else X_1.*AB-Chain*
X_0.*A-Chain*←if NP-trace(NP) then ↑NP
else X_1.*A-Chain*
condition:
¬ *Wh*-trace(NP) ∨ ¬X_1.*AB-Chain*
¬NP-trace(NP) ∨ ¬X_1.*A-Chain*

(b) X→X YP (YP≠NP)
attribution:
X_0.*AB-Chain*←if YP.*AB-Chain* then YP.*AB-Chain*
else X_1.*AB-Chain*
X_0.*A-Chain*←if YP.*A-Chain* then YP.*A-Chain*
else X_1.*A-Chain*
condition:
¬YP.*AB-Chain* ∨ ¬X_1.*AB-Chain*
¬YP.*A-Chain* ∨ ¬X_1.*A-Chain*

In (35a), independent attribution determines the functional type of the NP object. Setting to *nil* the *Chain* attribute of P, regardless of its type, captures concisely the fact that movement is never to object position (as implied by the θ-criterion). The attributes *A-Chain* and *AB-Chain* on the parent node X_0 are evaluated according to the type of the NP. Thus, if NP is *Wh*-trace, X_0.*AB-Chain* is a pointer to NP (↑NP); this signals the start of a nonargument chain. Otherwise, X_0.*AB-Chain* evaluates to the value of the corresponding attribute occurrence on the daughter node X_1. This value is non-*nil* if a nonargument chain extends across this node.

The use in (35) of an $\overline{X}$ scheme recursive on X allows for attachment of multiple complements of a head. This subtle point was ignored in the attribution for government and Case marking, but it is of interest here. The recursive scheme was introduced in (8) above. Its application is shown in (36), where the complements of the head verb are attached as sisters one at a time. The symbol X on the righthand side in (35a–b) is either the lexical head of the phrase defined, or a phrase consisting of the lexical head and several complements already attached. Upon lexical insertion of a formative under a category X, the attributes *AB-Chain* and *A-Chain* of X are set to *nil*—that is, no chain propagates below this node.

(36) (a) John$_i$ was [$_V$ [$_V$ given [ϵ]$_i$] a book]
(b) Who$_i$ did Paul [$_V$ [$_V$ give a book] to [ϵ]$_i$]
(c) What$_j$ was John$_i$ [$_V$ [$_V$ given [ϵ]$_i$] [ϵ]$_j$]
(d)*What$_j$ ... who$_i$ ... he [$_V$ [$_V$ gave [ϵ]$_i$] [ϵ]$_j$]

The attribute conditions in (35a–b) limit movement (of either kind, argument or nonargument) to only one extraction of each kind per phrase. By the first condition in (35a), for example, either NP is not a *Wh*-trace or there is no nonargument chain across the sister NP_1 node. Hence the examples (36a–c) are accounted for, but a configuration like (36d) may not be correctly attributed. An immediate consequence of this restriction on extractions is that the Chain rule does not account for parasitic gap constructions. The explanation of these constructions is controversial; we have not considered extension of the Chain rule to these cases.

The attribution for the productions that expand IP appears in (37). In (37a), the *Chain* attribute of the subject is immediately set to $\bar{I}$.*A-Chain*, which is either *nil* or points to a lower element of an A-Chain. Several interesting facts are implied at once by this simple assignment. NPs in subject position will never appear as intermediate elements of $\overline{A}$ chains. Since no condition is put on the functional type of this NP, it follows that PRO, overt NP, or *Wh*-trace may appear as head of an A-Chain. If there is no open A-Chain in the subtree dominated by $\bar{I}$ ($\bar{I}$.*A-Chain*=*nil*), this subject position will be the last element of a chain. The first attribute condition in (37a) limits to one the number of phrases that may be extracted from an IP phrase, per type of movement. The second condition, on external θ-role assignment by $\bar{I}$, blocks chains with

multiple θ-roles. The third condition requires a θ-role at the position or a link to an element with a θ-role. The condition rules out non-θ-marked $\overline{\text{A}}$ chains, as in *$What_i$ $[\epsilon]_i$ *seems that John is here*. The last two conditions thus contribute to the implementation of the θ-criterion.

(37) (a) IP→NP $\bar{\text{I}}$
attribution:
NP.*Chain*←$\bar{\text{I}}$.*A-Chain*
IP.*AB-Chain*←if *Wh*-trace(NP) then ↑NP
else $\bar{\text{I}}$.*AB-Chain*
IP.*A-Chain*←if NP-trace(NP) then ↑ NP
else *nil*
condition:
¬ *Wh*-trace(NP) ∨ ¬$\bar{\text{I}}$.*AB-Chain*
¬$\bar{\text{I}}$.*A-Chain* ∨ ¬$\bar{\text{I}}$.θ_E
$\bar{\text{I}}$.*A-Chain* ∨ $\bar{\text{I}}$.θ_E

(b) $\bar{\text{I}}$→I VP
attribution:
$\bar{\text{I}}$.*AB-Chain*←VP.*AB-Chain*
$\bar{\text{I}}$.*A-Chain*←VP.*A-Chain*

The attribution for the productions that expand CP is in (38). Note that by the first attribute condition in (38a–b), presence of C-specifier position is directly correlated to whether an $\overline{\text{A}}$ chain is propagated across the $\overline{\text{C}}$ node (value of $\overline{\text{C}}$.*AB-Chain*). The implications of the attribution may be studied with the help of the previous two sets of rules, (35) and (37).

(38) (a) CP→$\overline{\text{C}}$
attribution:
CP.*AB-Chain*←$\overline{\text{C}}$.*AB-Chain*
CP.*A-Chain*←$\overline{\text{C}}$.*A-Chain*
condition:
¬$\overline{\text{C}}$.*AB-Chain*

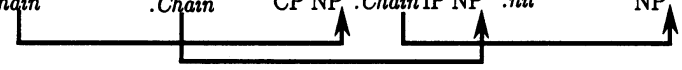
(b) CP→NP $\overline{\text{C}}$
attribution:
NP.*Chain*←$\overline{\text{C}}$.*AB-Chain*
CP.*AB-Chain*←if *Wh*-trace(NP) then ↑NP else *nil*
CP.*A-Chain*←$\overline{\text{C}}$.*A-Chain*

condition:
$\overline{\text{C}}$.*AB-Chain*
Wh-trace(NP) $\vee$ operator(NP)

(c) $\overline{\text{C}}\rightarrow$C IP
attribution:
$\overline{\text{C}}$.*AB-Chain*$\leftarrow$IP.*AB-Chain*
$\overline{\text{C}}$.*A-Chain*$\leftarrow$IP.*A-Chain*

The Chain rule just presented enforces attribute dependencies on the *Chain*, *A-Chain*, and *AB-Chain* attributes in S-structure trees. The attribution may be evaluated bottom-up in the trees defined by the grammar; *Chain* is inherited, but *A-Chain* and *AB-Chain* are synthesized, in contrast to the previous version of the rule (Correa, 1987). The operation of the rule is illustrated with the example (39) from Correa (1987).

(39) Who$_{.Chain}$ does John$_{.Chain}$ seem[$_{\text{CP}}$[$_{\text{NP}}$]$_{.Chain}$[$_{\text{IP}}$[$_{\text{NP}}$]$_{.nil}$ to love[$_{\text{NP}}$]$_{.nil}$]]



6. BARSS' CHAIN BINDING

The algorithm presented by Barss (1983) assumes the Linking theory of Higginbotham and is a representational characterization of nonargument chains. Rizzi (1986) presents a similar algorithm for argument chains. The framework assumed in this chapter is that of Chomsky (1981); thus we ignore the distinction between linking and coindexing (although in this section we use the linking terminology of Barss). Barss assumes that linking is not automatic under movement, and instead that free linking is carried out at S-structure. The representational characterization (40) of nonargument chains is a set of well-formedness conditions on the output of the free linking rule.[19]

(40) C is an $\overline{\text{A}}$-Chain if:

(i) $C = (a_1, \ldots, a_i, \ldots, a_n)$

(ii) a_n is θ-marked (or binds a θ-marked position)

(iii) a_i is in an $\overline{\text{A}}$ position, for all $i = 1, \ldots, n-1$

(iv) a_i links a_{i+1}, for $i = 1, \ldots, n-1$

(v) a_n is Case-marked; a_i is Caseless, for $i < n$

(vi) a_1 has semantic content

(vii) a_i is a *Wh*-trace, for $i = 2, \ldots, n$

(viii) For all i, j, category of a_i=category of a_j

This characterization of $\overline{\text{A}}$ chains is supplemented by the Locality Principle on Chains (LPC) (41). The notion *most local $\overline{A}$ position* in (41ii) is defined by Barss as in (42).

(41) Locality Principle on Chains (LPC):
Let $C = (a_1, \ldots, a_i, \ldots, a_n)$ be an $\overline{\text{A}}$- Chain. Then:

(i) a_i c-commands a_{i+1}, for $i = 1, \ldots, n-1$

(ii) a_i is the most local $\overline{\text{A}}$ position for a_{i+1}, for $i = 1, \ldots, n-1$

(42) X is *the most local $\overline{A}$ position* for Y if there is no $\overline{\text{A}}$ position Z (intervening between X and Y) such that Z is not in the same $\overline{\text{A}}$-Chain as a licensed position W, where W c-commands Y.

Scoping out the multiple negations in (42) we get the simpler (43).

(43) X is *a most local $\overline{A}$ position* for Y if for each $\overline{\text{A}}$ position Z (intervening between X and Y), there exists a licensed position W such that (i) Z is in the same $\overline{\text{A}}$-Chain as W, and (ii) W c-commands Y.

The Locality Principle on Chains is a close translation of Pesetsky's (1982) Path Containment Condition (PCC), not making use of the notion *path*. Condition (41ii) rules out overlapping but not nested *Wh*-movements, as in (44).

(44) (a) [What subject]$_i$ do you know [who$_j$ PRO to talk to e$_j$ about e$_i$]

(b) Who$_j$ do you know [[what subject]$_i$ PRO to talk to e$_j$ about e$_i$]

In (44a) the chains (*what subject*, e$_i$) and (*who*, e$_j$) satisfy the LPC since *what subject* and *who* are most local antecedent for e$_i$ and e$_j$, respectively. *Who* does not block the relation between *what subject*

and e_i since it is in the same $\overline{A}$ chain as e_j, which c-commands e_i.[20] (44b) on the other hand is not well-formed, since (*who*, e_i) does not satisfy the LPC. In spite of the close correspondence between the LPC and Pesetsky's PCC, they do not completely overlap. The two may be empirically distinguished, in favor of the LPC. The structures in question are those in which a nonargument position X intervenes between a *Wh*-trace and its antecedent. Consider (45a–b). The sentence is ambiguous, depending on the extraction site of the *Wh* element *when*. However, the sentence is disambiguated by an additional adverbial, as in (45c–d). In this case, the only interpretation is (45c), with the second adverbial attached to the lower clause. *Wh*-movement from the lower clause (45d) is blocked by presence of the adverbial in the higher clause.

(45) (a) When$_i$ did you tell [$_{CP}$ Mary to leave [ϵ]$_i$]

(b) When$_i$ did you tell [$_{CP}$ Mary to leave] [ϵ]$_i$

(c) When$_i$ did you tell [$_{CP}$ Mary to leave yesterday] [ϵ]$_i$

(d)*When$_i$ did you tell [$_{CP}$ Mary to leave [ϵ]$_i$] yesterday

The LPC accounts for (45d), since the last adverbial is the most local $\overline{A}$ position for the trace [ϵ]$_i$. Thus the chain (*when*, e_i) is ill-formed. The PCC on the other hand does not distinguish between (45c) and (45d) since there is only one $\overline{A}$ chain; it is satisfied trivially in both cases. The LPC may be incorporated into the Chain rule by constraining values of occurrences of the attribute *AB-Chain*. We omit the exact formulation of this constraint.

It may be verified that the chains defined by the Chain rule satisfy the Chain Binding conditions (40). The version of the rule defined here does not allow multiple *Wh*-extractions, so it trivially satisfies the PCC. See Correa (1987) for discussion of an extension that obeys Pesetsky's PCC. The LPC may be easily accounted for as noted above.

7. COMPLEXITY OF CHAIN CONSTRUCTION ALGORITHMS

Barss interprets the chain conditions (40) and the LPC as elements of a chain construction algorithm, acting as filters on the output of free linking. The architecture of the algorithm is parallel to that of the algorithm implicit in Chomsky's (1981) binding theory; it is a generate-and-test algorithm.

In spite of the possibility of interpreting the chain binding conditions as an algorithm, we propose instead that they should be interpreted as a *specification* of the output of any chain construction algorithm. It is extremely unlikely that a blind generate-and-test algorithm is embodied in an actual performance grammar. The reason for this is clear: the runtime complexity of a generate-and-test algorithm for chain construction is (unnecessarily) related exponentially to the length of the input string being analyzed.

This may be verified by a simple argument. First we need note that the number of nodes in a syntactic tree (say S-structure) is at most linearly related to the length of the terminal string dominated, provided the tree satisfies $\overline{\text{X}}$ theory conditions. The following argument shows under what conditions this claim is true. By the succession property of $\overline{\text{X}}$ grammars (Kornai, 1983; Pullum, 1985), the length of any derivation employing unit productions alone is limited to the maximum bar level of the grammar. This number is two under our assumptions, and hence the number of tree nodes introduced by such derivations alone, before a branching or terminal node is produced, is bounded by the bar level to two. If the tree in question does not have any empty categories on its frontier, the original assertion follows easily. For if the minimum degree of branching of the tree is $k > 1$, then the number of nodes required is bounded above by $(n+k-2)/(k-1)$. If unit productions are considered, but constrained by $\overline{\text{X}}$ theory, then this bound is increased at most by a constant multiplicative factor, namely the bar level.

This condition on empty categories does patently not hold in a Government-Binding theory grammar. The base component may generate infinitely many trees that ultimately yield the empty string. For example, the tree (46), obtained by expansion of the symbol CP without recursion on any category symbol, provides a basic recursive building unit. Recursion is via the symbol CP. All leaves of the unit (C, NP, I, V, and CP) may be expanded into the empty string. We advance as a substantive claim, however, that the number of empty categories in a well-formed (correctly attributed) tree is at most linearly related to the number of overt terminals dominated. The substantive fact about (46) is that other components of the grammar require that at least one of its leaves be overt. Typically a verbal element is required and it may appear under the V, I, or C node. If the verb is interpreted elliptically, then some other leaf—*e.g.*, an argument NP—must be overt.[21]

(46) $[_{CP} \ldots C\ [_{IP}\ NP\ [_{\bar{I}}\ [_{VP}\ V\ CP\ ]]]]$

Since each recursive unit (46) requires at least one overt element, the depth of the recursion in a structure generated from (46) is bounded by the number of overt elements generated. Furthermore, since the number of leaves in the unit is finite (bounded by the grammar), any structure built from (46) by recursion on CP will have a number of empty nodes bounded by the overt elements. Similar considerations may show that structures built by recursion on projections of lexical categories (NP, PP, and so forth) also have this property; typically an overt head is required. The reader is referred to Berwick and Weinberg (1984) for a similar argument to the one just presented. They show that the size of an annotated S-structure generated by a Government-Binding theory grammar is linearly related to the length of the input string.

Having established informally that the number of nodes in a tree generated by a GB grammar is at worst linearly related to the length of the terminal string dominated, the rest of the argument for the complexities of Chain Binding and the Chain rule follows easily. The cases of interest are those in which the number of (embedded) clauses present in the input is proportional to the length of the input. In these cases, and for fixed and small integer k, the number of NP nodes in a tree with $c \cdot n$ nodes is $c \cdot n/k$, where n is the length of the input. Assuming free linking, each NP node can be linked to any other NP node, or to none. Hence there are $c \cdot n/k$ possibilities for each NP, and the total number of candidate linkings generated is $(c \cdot n/k)^{c \cdot n/k}$ Assuming that the Chain Binding conditions (40) may be checked in constant time t_B for each candidate assignment, the running time of Chain Binding is $t_B \cdot (c \cdot n/k)^{c \cdot n/k}$, which is exponentially related to the length of the input string.

In contrast, the attribution associated with each production in the definition of the Chain rule may be evaluated in constant time. This may be verified by noting that the attribution associated with each production is strictly local, in the sense that it refers only to attribute occurrences in the elementary tree defined by the production. Furthermore, attribute values are not used to encode syntactic structure, so that the complexity of each value (*e.g.*, its size) is not related to input length and may be assumed to be constant.[22] Chain computation consists of the evaluation of the attributes *Chain*, *AB-Chain*, and *A-Chain* at each node in the tree and the conditions associated with them. Given

a surface tree, the number of productions needed to derive it is linearly related to the input length. It follows that Chain rule application is linearly related to the same string.[23]

In the discussion above we have ignored questions of ambiguity and parsing of the input surface string. Both chain algorithms were compared according to the amount of work each would have to do to identify chains in the input string, assuming a suitable surface structure is available.

8. GRAMMAR AS SPECIFICATION AND GRAMMAR AS REALIZATION

The contrast we have noted between the Chain Binding algorithm and the Chain rule raises some questions about the status of grammatical descriptions like the Chain Binding conditions (40), or a suitably reduced version of the same conditions, and other similar components of GB theory, including Move-α and the Binding theory. The components of the theory are assumed to be elements of the language-particular grammars that describe an ideal speaker's intrinsic competence. However, one of the claims often associated with a linguistic theory is that it provides a description of the actual structure of mentally represented grammars. That is, the theory is claimed to be psychologically real in a rather strong sense. Chomsky (1980, p. 187) remarks "There is no reason for the linguist to refrain from imputing existence to this initial apparatus of mind [universal grammar] as well".

We propose that descriptions of grammatical competence of the sort advanced by GB theory have no psychological reality—they do not describe the actual structure of mentally represented grammars. Instead, the description provided by the theory is an abstract and neutral *specification* of the grammatical competence of the native speaker. Indeed, Chomsky (1980, p. 187) adds to his remark above that "Proposed principles of universal grammar may be regarded as an abstract partial specification of the genetic program that enables the child to interpret certain events as linguistic evidence". The fact that is obscured by assigning psychological reality to a theory of competence is that there may be a wide gap between the "abstract partial specification" provided by the theory and the actual "genetic program" embodied in the speaker. The gap may exist even under all speaker idealizations assumed by the theory, and the specification can in fact have numerous genetic realizations.

We turn to this issue below.

The manner in which an abstract specification is realized in a performance grammar that meets observed performance constraints has not been addressed to a significant extent in the linguistic literature. Recent work on Government-Binding parsing by Stabler (1987), Fong (this volume), Johnson (this volume), and Abney and Cole (1986) has faced the problem of obtaining procedural interpretations of the grammatical devices assumed in Government-Binding theory. Stabler exploits the principles of Government-Binding theory to develop a grammatical formalism, restricted logic grammar, suited especially for the statement of natural language grammars. He remarks on the problem of mapping from the abstract principles in the theory to a procedural interpretation "It is unfortunate that efficient implementation of these constraints [on movement] requires such careful attention to procedural details" (1987, p. 8).

Johnson axiomatizes part of GB grammar in first-order logic, and applies logic program transformation techniques (Tamaki and Sato, 1984) to obtain restatements of the grammar for which the procedural interpretation given by a Prolog interpreter yields a prototype parser. Thus, program transformations bridge with partial success the gap between the original specification and the grammar embodied in the runtime program. Abney and Cole use a model of distributed computation to formulate parsing procedures that satisfy the grammatical principles of Government-Binding theory. It appears that the work noted focuses on the interpretive derivation of syntactic representations that are transformationally defined in Government-Binding theory. That is, traces are hypothesized and coindexed with antecedents without using Move-α directly. The Chain rule differs from that work since the rule is a declarative definition of chain relations at S-structure. The procedural interpretation of the rule is assigned independently by an attribute evaluator.

The distinction between grammatical descriptions intended as language specifications and descriptions intended as more realistic models of how grammar is put to use in generation and understanding is seen clearly in the following example. Consider the *sorting language* S of strings over some finite subset $\mathbb{N}$ of the natural numbers. Each string has the property that it may be split into two substrings x and y, such that y is the sorted version of x:

$$(47)\quad S = \{xy : x, y \in \mathbb{N}^* \ \wedge \text{sorted}(x, y)\}$$

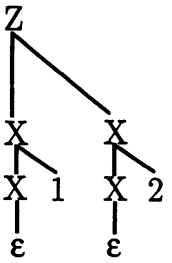
This simple language might be given a phrase structure description (48). A sample parse tree for the string '1 2' is shown in (48d). The left-recursive productions (48b–c) define the 'concrete syntax' that would commonly be given to a production $Z \rightarrow X^* \; X^*$ employing regular expression notation.

(48) (a) $Z \rightarrow X \; X$

(b) $X \rightarrow \epsilon$

(c) $X \rightarrow X \; t$, for each $t \in \mathbb{N}$

(d)



The phrase structure rules as defined clearly overgenerate; they generate $\mathbb{N}^*$. To see this, note that each string s in S may be split into two substrings x and y. The first constraint we may observe is the Ordering Constraint (49) on the second half of the string:

(49) Ordering Constraint (OC):
$*s = xy$, unless y is ordered,
where $x = x_1 \ldots x_n$ is *ordered* if
$i < j$ implies $x_i \leq x_j$, for all $i, j = 1, \ldots, n$.

The further constraint (50) yields the language observed.[24]

(50) Permutation Constraint (PC):
$*s = xy$, unless y is a permutation of x,
where x is a *permutation* of y if
(i) $x = y = \emptyset$, or
(ii) for each $x_i \in X$, $x_i \in y$, and
$x - x_i$ is a permutation of $y - x_i$.

The grammar consisting of (48), (49), and (50) is an abstract and concise description of the intended language. The constraints of the grammar, OC and PC, may be interpreted as components of an abstract algorithm for the generation and analysis of strings in the language.

However, it would be foolish to use the abstract algorithm for a practical purpose or, if the language defined were a natural one, to claim *a priori* that those principles are elements of a model for how strings in the language are processed by speakers of the language.

The principles (49) and (50) have the features of a good specification: they are precise, consistent, complete, and unbiased towards a particular realization of the sorting relation they define. Refinement of the specification (Jones, 1980) yields a realization or particular algorithm whose only relation to the specification is, ideally, logical equivalence. The sorting relation may be refined in several ways, to yield one of the known sorting algorithms. Performance is likely to be one of the criteria used in the selection of one refinement over another. The refinement may be done systematically, by applying one of several specification transformation schemes. Thus a sort-by-permutation logic program (Clocksin and Mellish, 1981) may be transformed into a more efficient form by application of the fold/unfold transformation (Tamaki and Sato, 1984). Johnson (this volume) uses this approach in the derivation of the more efficient of his PAD parsers. Typically, however, specifications must be refined in other creative ways, which follow no particular systematic pattern. The Quicksort algorithm of Hoare (1962) is an interesting example.

As the example of the sorting language S illustrates clearly, there may be a wide gap between the description of some linguistic phenomena and the mechanism that actually produces the phenomena. Marr (1977) argues that both levels of description, specification of what is to be computed and the particular algorithm that implements the computation, are important for a complete understanding of an information processing problem.[25] This calls for a distinction on principles of grammar, according to whether they may be taken only as *specifications* of linguistic phenomena (their primary function), or also as natural algorithms for dealing with the phenomena. It seems that only the former is the proper interpretation of the elements of grammar assumed in Government-Binding theory. From our point of view, the psychological claim that may be made about universal grammar is that it imputes existence to a mechanism that satisfies the abstract specification given, not to a mechanism with the 'specific internal organization' postulated. Questions about the 'physical reality' of the principles of a grammatical theory demand a precise procedural interpretation to be associated with them, and may be answered only by correlation of psycholinguistic ob-

servations with the predicted behavior of the theory for actual inputs.

9. CONCLUSION

This chapter has presented an interpretive Chain formation rule (*Chain rule*) and defined it explicitly in the formalism of attribute grammar. The formulation of the rule was influenced to a large extent by the rule Move-α and several other components in Government-Binding theory, including the thematic, Case, and government subtheories. While the Chain rule did not include an account of parasitic gaps and multiple *Wh*-extractions from a given phrase, it seems likely that the rule can be refined to cover all cases of trace-antecedent relation currently explained by Move-α. By contributing towards a theory a performance, the Chain rule argues indirectly for an interpretive theory of grammar. We note, however, that the work of Koster (1978), Rizzi (1986), and Barss (1983) also suggests an advantage of the interpretive approach over the transformational one, on grounds of descriptive and explanatory adequacy.

The principles of grammar assumed in Government-Binding theory are best seen as specifications of linguistic phenomena, and not as algorithms that deal with the phenomena. A significant gap exists between the principles given by the theory and the actual algorithms that may be used in human sentence processing. This view is consistent with the goal of linguistic theory to provide an abstract and neutral characterization of the knowledge of language possessed by native speakers.

ACKNOWLEDGEMENTS

Thanks are due to Robert C. Berwick for detailed comments on an earlier version of this article, Susumu Kuno for the reference to Barss (1983), and the Natural Language Processing group at IBM Research. Any shortcomings and inaccuracies are of course my own.

NOTES

[1] We assume intermediate traces left by application of Move-α are retained at S-structure. If traces are optionally left by Move-α, as in Lasnik and Saito (1984), then Subjacency must be taken as a condition on derivations.

[2] We identify I with the category AUX(iliary) of Steele *et al.* (1981).

[3] Notice that this changes the account that may be given of *that*-trace phenomena in terms of c-command. The C-specifier position c-commands subject position (I-

specifier), regardless of whether an overt complementizer element is dominated by C.

[4] Interest in the structural properties of natural language provided empirical motivation for the study of linguistic devices with more generative power that regular grammars and more special structure than unrestricted rewriting systems. Chomsky (1959) dismissed unrestricted grammars as devices of linguistic interest, since there is no adequate way to define for them a method for obtaining structural descriptions of the strings they generate.

[5] We write $X.a$ to denote *attribute occurrences* of attributes of symbol X. Thus $X.a \neq Y.a$, if $X \neq Y$, for all categories X and Y. By this convention, we distinguish the attribute occurrence *Case* associated with V (V.*Case*) from the attribute occurrence *Case* associated with NP (NP.*Case*).

[6] This intended characteristic of attribute grammars cannot be enforced in a vacuum, ignoring the *types* of attributes allowed. For when pointer types are permitted, as in the PLNLP system of Heidorn (1972), attribution functions may refer to attributes of nodes arbitrarily far away from the node of the attribute being defined.

[7] We do not impose any restriction on the degree of ambiguity of the underlying grammar G. In fact, it is easy to verify that the language-particular base grammars implied in current GB theory are infinitely ambiguous.

[8] Knuth (1968) shows that restriction of attribute grammar to just synthesized attributes does not change the class of attributions that may be defined on derivation trees or its formal descriptive power.

[9] If the implementation medium for attribute evaluation provides *unification* (Robinson, 1965), then evaluation of an attribution rule need not coincide with instantiation of the attribute value the rule defines. In general, evaluation of the attribution rule does not require knowing the values of all input arguments to the rule.

[10] This definition assumes *value semantics* for attribute evaluation.

[11] In the PLNLP language of Heidorn (1972), tree nodes are represented by record structures encoding the categorial information of the node and the attributes associated with it. In spite of the use of pointers, the attribution is strictly local. Each pointer value may be assumed to be a runtime-generated index, with the convention that a node's *node number* is the address at which the node is stored. The up-arrow notation in pointers does not bear any relation to the same notation in lexical-functional grammar.

[12] This partitioning of empty categories, well established in early GB literature, is questioned in more recent work. Aoun (1985), for example, proposes that a variable is both referential and anaphoric. This cross-classification cannot be achieved in terms of the attributes *anaphoric* and *pronominal* assumed in the earlier work.

[13] This distinction is not new. In Lasnik and Uriagereka (1988, p. 71), intermediate traces "are presumably not any of the empty categories we are considering [NP-trace, *Wh*-trace, and PRO]", and they bear no *anaphoric* and *pronominal* attributes. This view implies an elaboration of the transformational component since, among other things, it is now necessary to specify for each transformation which attributes of a moved phrase it retains and which it leaves. Within the attribute grammar framework assumed here, the set $A(X)$ of attributes associated with each category X is fixed.

[14] A closely related alternative is to associate the attribution in (29) with each

production that has NP on the righthand side. This increases the size of the grammar, since the attribution is replicated in all such productions, but does not change the amount of computation required to determine functional type, compared with (29).

[15] As with the attribution (29), it is possible to move the attribute conditions in (30) to all productions with NP on the righthand side.

[16] Due to the use of pointer types rather than integers, the Chain rule presented here defines *linking* relations from antecedent to traces. This is an inessential feature of the rule, though. The coindexing relation assumed in Chomsky (1981) may be obtained by computing the reflexive and transitive closure of linking. Correa (1987) uses coindexation explicitly, but requires explicit enumeration of tree nodes.

[17] An exception exists, depending on the acceptability of expletive chains, as in *It$_i$ seems* [ϵ]$_i$ *to be likely that John won the race.*

[18] These two attributes should make accessible also categorial, agreement, and Case information of the element they point to.

[19] Barss does not intend (40) as a definition of $\overline{\text{A}}$ chains; some properties may follow from other principles in the grammar.

[20] Barss assumes reanalysis of the verb and adjacent preposition, so that e_j is object of the reanalyzed verb.

[21] This is a substantive claim for which I have no formal proof.

[22] Strictly speaking, we must allow that the representation of node indices or pointer values will grow logarithmically with the length of the input string. The point made is that our use of attribute-value structures differs significantly from the use of the same objects in other formalisms such as lexical-functional grammar. The present use is more consistent with the notion of *complex symbol* introduced in Aspects-type grammars.

[23] Taking into account the representation of pointers, evaluation of the Chain rule requires $O(n \cdot \log(n))$ steps.

[24] The Permutation Constraint subsumes a weaker constraint, the Equal Length Constraint (ELC), which asserts that a string s is not in the language S unless it splits into two substrings of equal length.

[25] I thank Bob Berwick for this observation.

REFERENCES

Abney, S. and J. Cole: 1986, 'A Government-Binding Parser', unpublished manuscript, Department of Linguistics and Philosophy, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Aoun, J.: 1985, *A Grammar of Anaphora*, MIT Press, Cambridge, Massachusetts.

Barss, A.: 1983, 'Chain Binding', unpublished manuscript, Department of Linguistics and Philosophy, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Berwick, R. and A. Weinberg: 1984, *The Grammatical Basis of Linguistic Performance*, MIT Press, Cambridge, Massachusetts.

Chomsky, N.: 1959, 'On Certain Formal Properties of Grammar', *Information and*

Control **2**, 137–167.

Chomsky, N.: 1970, 'Remarks on Nominalization', in R. Jacobs and P. Rosenbaum (eds.), *Readings in English Transformational Grammar*, Ginn and Co., Waltham, Massachusetts, pp. 184–221.

Chomsky, N.: 1980, *Rules and Representations*, Columbia University Press, New York.

Chomsky, N.: 1981, *Lectures on Government and Binding: The Pisa Lectures*, Foris, Dordrecht, Holland.

Chomsky, N.: 1982, *Some Concepts and Consequences of the Theory of Government and Binding*, MIT Press, Cambridge, Massachusetts.

Chomsky, N.: 1986, *Barriers*, MIT Press, Cambridge, Massachusetts.

Clocksin, W. and C. Mellish: 1981, *Programming in Prolog*, Springer-Verlag, New York.

Correa, N.: 1987, 'An Attribute Grammar Implementation of Government-Binding Theory', *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, California, pp. 45–51.

Emonds, J.: 1976, *A Transformational Approach to Syntax*, Academic Press, New York.

Heidorn, G.: 1972, *Natural Language Inputs to a Simulation System*, Naval Postgraduate School, Technical Report No. NPS-55HD72101A, Alexandria, Virginia.

Hoare, C.: 1962, 'Quicksort', *Computer Journal* **5**, 10–15.

Irons, E.: 1961, 'A Syntax-directed Compiler for ALGOL 60', *Communications of the Association for Computing Machinery* **4**, 51–55.

Jackendoff, R.: 1977, $\overline{X}$ *Syntax: A Study of Phrase Structure*, MIT Press, Cambridge, Massachusetts.

Jensen, K.: 1986, 'Binary Rules and Non-binary Trees: Breaking down the Concept of Phrase Structure', in A. Manaster-Ramer (ed.), *Mathematics of Language*, John Benjamins, Amsterdam, pp. 65–86.

Johnson, M.: this volume, 'Parsing as Deduction: The Use of Knowledge of Language', pp. 39–64.

Jones, C.: 1980, *Software Development: A Rigorous Approach*, Prentice-Hall International, Series in Computer Science, Englewood Cliffs, New Jersey.

Kastens, U., B. Hutt, and E. Zimmermann: 1982, *GAG: A Practical Compiler Generator*, Lecture Notes in Computer Science, Springer-Verlag, New York.

Knuth, D.: 1968, 'Semantics of Context-free Languages', *Mathematical Systems Theory* **2**, 127–145.

Kornai, A.: 1983, '$\overline{\text{X}}$ Grammars', in J. Demetrovics, G. Katrona, and A. Salomaa (eds.), *Algebra, Combinatorics, and Logic in Computer Science*, vol. 2, North-Holland, Amsterdam, pp. 523–536.

Koster, C.: 1971, 'Affix Grammars', in *IFIP Working Conference on Algol 68 Implementation*, North-Holland, Amsterdam, pp. 95–109.

Koster, J.: 1978, 'Conditions, Empty Nodes, and Markedness', *Linguistic Inquiry* **9**, 551–593.

Lasnik, H. and J. Uriagereka: 1988, *A Course in GB Syntax: Lectures on Binding and Empty Categories*, MIT Press, Cambridge, Massachusetts.

Lasnik, H. and M. Saito: 1984, 'On the Nature of Proper Government', *Linguistic Inquiry* **15**, 235–289.

Marr, D.: 1977, 'Artificial Intelligence: A Personal View', *Artificial Intelligence* **9**, 37–48.

Pesetsky, D.: 1982, *Paths and Categories*, Ph.D. dissertation, Department of Linguistics and Philosophy, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Petrick, S.: 1965, *A Recognition Procedure for Transformational Grammars*, Ph.D. dissertation, Department of Linguistics, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Pullum, G.: 1985, *Assuming Some Version of the $\overline{X}$ Theory*, Syntax Research Center, University of California at Santa Cruz, Santa Cruz, California.

Rizzi, L.: 1986, 'On Chain Formation', in H. Borer (ed.), *The Grammar of Pronominal Clitics—Syntax and Semantics, vol. 19*, Academic Press, New York, pp. 65–95.

Robinson, A.: 1965, 'A Machine-oriented Logic Based on the Resolution Principle', *Journal of the Association for Computing Machinery* **12**, 23–41.

Sharp, R.: 1985, *A Model of Grammar Based on Principles of Government and Binding*, M.S. dissertation, Department of Computer Science, University of British Columbia, Vancouver, British Columbia.

Shieber, S.: 1986, *An Introduction to Unification-based Approaches to Grammar*, CSLI Lecture Notes No. 4, University of Center for the Study of Language and Information, Stanford, California.

Stabler, E. P. Jr.: 1987, 'Restricting Logic Grammars with Government-Binding Theory', *Computational Linguistics*, **13**, 1–10.

Steele, S.: 1981, *An Encyclopedia of AUX: A Study in Cross-linguistic Equivalence*, MIT Press, Cambridge, Massachusetts.

Stowell, T.: 1981, *Origins of Phrase Structure*, Ph.D. dissertation, Department of Linguistics and Philosophy, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Tamaki, H. and T. Sato: 1984, 'Fold/Unfold Transformation of Logic Programs', *Proceedings of the Second International Logic Programming Conference*, Uppsala University, Uppsala, Sweden, pp. 127–138.

Van Riemsdijk, H. and E. Williams: 1986, *An Introduction to the Theory of Grammar*, MIT Press, Cambridge, Massachusetts.

Van Wijngaarden, A. (ed.): 1969, 'Report on the Algorithmic Language ALGOL 68', *Numerical Mathematics* **14**, 79–218.

Waite, W. and G. Goos: 1984, *Compiler Construction*, Springer-Verlag, New York.

Watt, D. and O. Madsen: 1983, 'Extended Attribute Grammars', *The Computer Journal* **26**, 142–153.

Departamento de Ingeníeria Eléctrica, W-311,
Universidad de los Andes,
Apartado Aéreo 4976,
Bogotá, D.E., Colombia