INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book. These are also available as one exposure on a standard 35mm slide or as a 17" x 23" black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 8914562

Syntactic analysis of English with respect to government-binding grammar

Correa, Nelson, Ph.D.

Syracuse University, 1988

Copyright ©1988 by Correa, Nelson. All rights reserved.

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

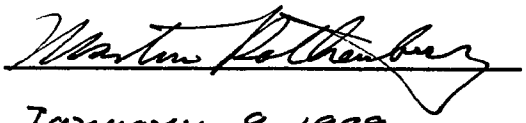
Syntactic Analysis of English with respect to Government-binding Grammar

by

Nelson Correa
B.S., Universidad Javeriana, Bogotá, Colombia, 1979
M.S., Syracuse University, 1981
M.A., Syracuse University, 1985

Abstract of Dissertation

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering in the Graduate School of Syracuse University
December 9, 1988

Approved 

Date January 9, 1989

Abstract

An attribute-grammar formulation of the Government-binding (GB) theory of natural language is proposed in which attribute definitions capture the substantive aspects of transformations and most component subtheories in Government-binding. Attributed definitions are used in place of transformational rewriting to formulate an interpretive *Chain rule*, with similar empirical effects to the transformation move-α in the theory. Attribute-grammar provides a precise computational framework from which a practical analysis procedure for a subset of English is systematically developed. It is thus shown that attribute grammar is a framework suitable for the description of natural language and for the investigation of performance models for the same.

The principal claim associated with this thesis is that the attribute grammar (AG) formalism is sufficient and of major interest for the description of natural language. This claim is proved in two steps. First, it is shown that under even a trivial kind of attribution domains and functions, attribute grammars have the expressive power of Type-0 grammars. Hence, the languages that can be described by the kind of linguistic devices assumed in current GB theory, can also be described within the simpler and more uniform framework of attribute grammar. Although attribute grammars have the same generative power as Turing machines, they are of major linguistic interest, in the sense that generation of a string automatically defines relevant structural information.

The second and more substantive part of the defense of our claim involves an AG specification of English, assuming the notions and principles of the Government-binding theory. We provide, for each component of the GB theory, its counterpart in the AG specification. Of particular interest is the abandonment of transformations and the use instead of the interpretive Chain rule for the definition of trace-antecedent relations in derivation trees. We show that the various linguistic constraints that movement structures are subject to, including the Subjacency and Path Containment conditions, are easily embodied in the formulation of the interpretive rule. In addition to the linguistic advantages of the Chain rule, including simplification of the

grammatical model assumed, an important advantage of the rule over the transformational theory is that the former may be applied effectively to compute trace-antecedent relations in a given surface string.

The feasibility of the use of attribute grammar for models of linguistic performance is shown by producing an extended LL(1) parser and attribute evaluator which constitute a practical analysis procedure for a subset of English. The analysis procedure is attribute-directed, effectively combining LL(1) derivation steps with attribute evaluation. The procedure is thus is able to cope with the extreme ambiguity of the underlying context-free syntax. Attribute evaluation uses unification as a basic operation and makes limited use of a generate-and-test approach for the evaluation of certain attributes. The procedure by which the attribute evaluator is derived from the grammar is described in detail.

Syntactic Analysis of English with respect to Government-binding Grammar

by

Nelson Correa
B.S., Universidad Javeriana, Bogotá, Colombia, 1979
M.S., Syracuse University, 1981
M.A., Syracuse University, 1985

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering in the Graduate School of Syracuse University
December 9, 1988

Approved 

Date 1/9/89

Copyright 1988

Nelson Correa

Acknowledgements

I acknowledge with gratitude the assistance that has been provided to me by the chairman and members of my dissertation committee, Professors Martin Rothenberg; Jaklin Kornfilt; Susumu Kuno; William Purdy; and Edward Stabler. Special thanks are due to Professor Kornfilt for constant encouragement and support, both at a linguistic and personal level, and to Professor Kuno for several helpful discussions and for agreeing to be an outside member of the committee.

I would also like to thank Dr. George E. Heidorn, Dr. Karen Jensen, and the members of the Natural Language Processing group at IBM Research for their support during the last stage of writing the dissertation and for a stimulating environment in which the research presented here could be continued.

Robert C. Berwick, Mark Johnson, Judy Kegl, Mary Laughren, and Beth Levin provided sporadic but helpful and encouraging discussions on aspects of this research at various times.

I also thank Bernardita Calinao, my parents, family, and friends for their constant support and encouragement.

Finally, I thank the Colombian Fulbright Commission, which made my graduate studies possible in the first place. This work was supported by Teaching Assistantships from the Department of Electrical and Computer Engineering and Research Assistantships from the New York State CASE Center at Syracuse University.

Contents

Chapter 1. Introduction

The objective of linguistic theory is to characterize natural language, assuming several idealizations about language and speakers. Two aspects are traditionally considered in the description of language. One of these is the definition, for each particular language, of some device or *generative grammar* that can effectively enumerate the sentences in that language. The second concerns the assignment of a structural description to each sentence generated. The structural description of a sentence captures information about how the sentence is understood by speakers of the language and how it is used. This information indicates, for example, whether the sentence is ambiguous and is a basis for studying the structural similarity of the sentence to other sentences in the language. The information concerning the structure of sentences must conform to the linguistic intuition of native speakers. A grammar is said to be *descriptively adequate* to the extent that it correctly characterizes the intuitions of the native speaker.

A higher level of adequacy on linguistic theory may be imposed by considering two additional empirical problems. The first is the problem of language acquisition. Given that knowledge of particular languages is not innate in native speakers, a linguistic theory should explain how this knowledge may be acquired by a native speaker. Let us call this knowledge the *competence* of the speaker. The second problem concerns the use of language. Given that the grammatical devices used for linguistic description are of a fairly abstract nature, with no obvious relation to analytic or generation procedures, linguistic theory should explain the manner in which knowledge of language is put to use in the production and understanding of utterances. We shall refer to the ability of a speaker to use language as the *performance* abilities of the speaker. A linguistic theory is said to be *explanatorily adequate* to the extent that it provides adequate theories of acquisition and performance for the class of devices it defines.

A linguistic theory is of interest to the extent that it contains a correct characterization of the *class* of descriptively adequate devices that enumerate particular natural languages. If the class of languages to be characterized is finite, the class of devices that generate them can in principle

be listed. A more insightful and useful approach, however, is to provide a grammatical *model* that the devices satisfy. Empirical considerations of adequacy are often crucial in deciding that the class of devices should be defined in one particular way rather than other.

Linguistic theory makes three important idealizations about natural language. First, it assumes that a natural language is a set of strings which can be effectively listed, and which have certain structure and properties associated with them. In this abstraction, the communicative function and the context in which language arises are ignored (Cherry, 1978). Second, it is implicit in the assumptions that the set of strings that constitute a natural language can be precisely characterized. Grammaticality can then be defined as membership in the set. This view abstracts from the fact that grammaticality judgements for a given utterance are sometimes fuzzy and not constant across speakers of a given language. Further, any device of linguistic interest generates an infinite set of strings; yet, at any given time, the number of sentences that has been uttered is finite. This introduces another aspect of idealization; certain strings which have and may never be uttered are defined to be in the language. Last, linguistic theory abstracts from performance factors in language use. A simple case involves sentence length. A set of strings over a finite vocabulary can be infinite only if its sentences can be arbitrarily long (but still finite in length). Performance factors and situation of use limit sentence length to modest numbers.[1] Other more interesting cases of performance limitations may be seen in recursive center embedding of utterances.

In this study we adopt the above idealizations about language. Summarizing, we assume that the main goal of linguistic theory is to describe the linguistic competence of an ideal native speaker; that is to say, what the speaker knows in some abstract sense, ignoring the function that knowledge of language serves, its context, or the manner in which it may actually be put to use. To achieve this goal, linguistic theory is free to develop and avail itself of any formal devices that are found appropriate for the description of language. To attain explanatory adequacy, however, a linguistic theory must produce acquisition and performance models for the class of linguistic devices it assumes. The models show how a competence grammar can be acquired and put to use in language production and understanding. The performance model, in particular, explains the remarkable facility with which knowledge of language is put to use by actual speakers. This

---

[1] Beale (1985) reports that in a sample of the Lancaster-Oslo/Bergen Corpus of British English with about 35,000 token words there are 1,500 sentences, giving an average sentence length of near 23 words per sentence.

requirement on linguistic theory is only a complementary and natural extension of the usual learnability constraints on a theory to attain explanatory adequacy (Chomsky, 1981). It imposes some further restrictions on the class of devices that are made available to the theory.

1.1 Problem statement and approach

The research presented here extends the Government-binding (GB) theory of natural language formulated in (Chomsky, 1981, 1982). Government-binding theory falls within the framework of the so-called Extended Standard Theory of transformational generative grammar. An argument is made here for a particular and highly explicit version of the theory. The way in which the theory is extended is by formulation of a performance model for it. The development of models of performance for Government-binding has been largely unchartered territory thus far. Most research has been concerned with the development of the theory of competence and models of language acquisition. As a result, the theory is perhaps the most advanced of the currently existing linguistic theories, but it does not offer clues as to how the competence grammars described by the theory are put to use in language production and analysis. Government-binding theory resorts to a variety of partially formalized grammatical devices, with no obvious relation to analysis or generation procedures. The obstacles to be overcome here are twofold: (i) one of formalization of the statement of the theory and (ii) one of relating the devices assumed by the theory to analysis and generation procedures.

The need for formalization in Government-binding theory is a pressing one, as pointed out by (Barton, Berwick, and Ristad, 1987, p. *x*), who examine computational complexity questions for several current linguistic theories, but omit discussion of Government-binding for lack of an adequate formalization. Chomsky (1981, p. 336) remarks that "there is sufficient depth and complexity of argument so that formalization will not merely be a pointless technical exercise but may bring to light errors or gaps and hidden assumptions, and may yield new theoretical insights and suggest new empirical problems for investigation."

We will overcome the two obstacles noted, particularly (ii), by introducting a distinction between competence grammar and performance grammar and postulating that the relation between the

two is one of a formal specification to a refinement.[2] In this way, the goal of the theory of competence is the characterization of the class of competence grammars. A theory of performance, on the other hand, should characterize the class of performance grammars, establish the correspondence between competence and performance, and provide a method by which performance grammars may be related to parsing and generation automata. A priori, it seems reasonable to ascribe psychological reality only to performance grammars; a competence grammar is only an abstract description of the performance grammar with, possibly, no physical embodiment. Thus, the goal of a theory of language acquisition is to explain the manner in which a performance grammar may be acquired, on the basis of external linguistic data.

This research shows that the formalism of attribute grammar, introduced by (Knuth, 1968) in the context of programming languages, is quite sufficient and adequate for the statement of performance grammars for Government-binding theory. We will remain ambiguous throughout the work as to whether attribute grammar is an adequate tool for formalization of the theory of competence in Government-binding. An attribute grammar is defined by an underlying context-free grammar, which is extended by associating attribute fields with each symbol in the underlying grammar. The attribute fields of a symbol are used to represent context-sensitive and semantic properties of the symbol. The syntactic representations that an attribute grammar defines are the derivation trees defined by the context-free grammar, but decorated with attribute fields at each node. The attribute fields are evaluated according to attribution statements made in the grammar. These statements have the property that each is associated with a specific production in the grammar. Attribute grammar is a precise and well-understood formalism for the characterization of the syntax and semantic of languages. We address the problem of formalization noted above, if only for the theory of performance, by adoption of this formalism.

The difficulty of relating the linguistic devices assumed in Government-binding theory to analysis and generation procedures stems from two properties of the current theory. One of these is the use of transformational operations in the statement of competence grammars. The second concerns the so-called "principle-based approach" to the definition of syntactic representations.

---

[2] The notions of "specification" and "refinement" are used here in the sense in which they are used in the area of program development (Jones, 1980). Formal specifications are non-procedural in nature and declarative. A good specification is precise, consistent, and unbiased towards a particular realization. A refinement satisfies the input-output relation defined by the specification, but it may be stated in a more primitive and procedural language, and may contain many details of realization not present in the original specification.

These two technical obstacles in the theory are overcome by, first, adopting a simplified grammatical formalism in which grammatical transformations are eliminated and, second, by refinement of the system of principles, which in the current theory are statements over complete syntactic representations, to statements or principles whose domain is elementary trees. Attribute grammar is thus an appropriate language for the refinement.

Transformations are generalized rewriting operations on tree domains. Although Government-binding uses a very restricted class of transformations in its definitions, essentially a class of transformations restricted to the rule move-α (Chomsky, 1981), previous research has not succeeded in producing a method to reconstruct transformational derivations, given an arbitrary input string. In fact, the work of (Kimball, 1967) and (Peters and Ritchie, 1973) shows that the problem is in general undecidable. Previous recognition procedures for transformational grammar by (Matthews, 1962) and (Petrick, 1965) are analysis-by-synthesis procedures, in which surface structures are hypothesized, and inverse transformational mappings are applied to test hypothetical deep structures against the class of deep structures. A more recent procedure by (Sharp, 1985), designed specifically for Government-binding theory, is also an analysis-by-synthesis procedure. It, however, takes advantage of the more constrained model of the current theory, specifically the X-Bar theory and the restricted class of transformations, move-α.

The principled-based approach in Government-binding grammar divides the grammar into two components, a set of rewrite and annotation *rules* for the generation of annotated syntactic representations, and a set of constraints or *principles* on the representations produced and on the application of the rules. This approach to grammar is formulated in (Chomsky and Lasnik, 1977). The set of rules in the grammar is extremely general and unconstrained. Rules are, furthermore, free to interact with each other. As a result, the rule system overgenerates. The role of principles in the grammar is to constrain the class of representations that may be generated to the empirically adequate set. Examples from the rule component in GB grammar include the base rules and the sets of transformations, including move-α. These rules define a system which, among other properties, is infinitely ambiguous; it defines infinitely many derivations for the empty string. The system of principles includes the so-called Case filter and the binding axioms, which will be discussed in Chapter 2.

1.2 Devices for formal linguistic description

The devices used for natural language description are string rewriting systems or *generative grammars.* Let V be a finite vocabulary of symbols, which can be partitioned into two subsets N and Σ of "categories" and "terminal" (or input) symbols. A generative grammar on V is a finite collection of rewriting rules of the form $\phi \rightarrow \psi$, where ϕ and ψ are strings of symbols over V, and ϕ is non-empty. Given a string $\gamma = \alpha\phi\beta$ of symbols over V, the rule $\phi \rightarrow \psi$ may be used to rewrite γ as $\gamma' = \alpha\psi\beta$. We write $\gamma \Rightarrow \gamma'$, to indicate that γ *directly derives* γ', by application of a rule in the grammar. If, given two strings α and β over V, there exists strings $\gamma_1, ..., \gamma_n$ such that $\alpha = \gamma_1$, $\beta = \gamma_n$, and $\gamma_i \Rightarrow \gamma_{i+1}$, for $i = 1, ..., n\text{-}1$, we write $\alpha \Rightarrow^* \beta$, and say that α *derives* β.

A generative grammar identifies a special or *initial* symbol Z of N. Given this definition, the *language* generated by the grammar can be defined as the set of strings over Σ which can be derived from the initial symbol Z. That is, if G denotes the grammar, L(G) its language, and Σ^* the set of strings over Σ, L(G) is the set $\{ \omega : \omega \in \Sigma^* \text{ and } Z \Rightarrow^* \omega \}$.

The class of languages that may be produced by a generative grammar is exactly the class of languages whose sentences may be effectively enumerated by a Turing machine. Chomsky (1959) studied a sequence of restrictions on rewriting systems that limit the grammars first to two types of phrase structure grammar, and then to finite automata. The following restrictions were considered:

(1) R.1. If $\phi \rightarrow \psi$ is a rule, then there exist $A \in N$ and $\phi_1, \phi_2, \omega \in V^*$ such that $\phi = \phi_1 A \phi_2$, $\psi = \phi_1 \omega \phi_2$, and $\omega \neq \epsilon$

R.2. If $\phi \rightarrow \psi$ is a rule, then there exist $A \in N$ and $\omega \in V^*$ such that $\phi = A$, $\psi = \omega$, and $\omega \neq \epsilon$

R.3. If $\phi \rightarrow \psi$ is a rule, then there exist $A, B \in N$ and $a \in \Sigma$, such that $\phi = A$ and $\psi = aB$ or $\psi = a$.

The symbol 'ϵ' in (1) is used ambiguously to denote the empty string and set-membership. The empty string 'ϵ' is a member of Σ^*, for any Σ. Situation of use of the symbol is in each case sufficient to obtain the intended meaning. For $i = 1, 2, 3$, we say that G is a grammar of *type-i* if

it meets restriction R.*i*. A type-0 grammar is one whose rewrite rules $\phi \rightarrow \psi$ are unrestricted. This classification is known as the *Chomsky hierarchy* of grammars and is carried over to languages, so that L(G) is a *type-i* language if G is a type-*i* grammar. The interest in studying these restrictions is in their use to define the class of devices from which the grammars of particular languages could be drawn. Restriction R.3 limiting grammars to finite automata was shown to be too strong; these grammars will not contain the grammar of English (Chomsky, 1957). Chomsky (1959), on the other hand, dismisses type-0 grammars as devices of linguistic interest, since there is no adequate way of obtaining a structural description for the strings generated by such systems.

In (Chomsky, 1959, p. 139), "interest in structural properties of natural language thus serves as an empirical motivation for the investigation of devices with more generative power than finite automata, and with more special structure than Turing machines." Chomsky (1957) argues that grammars meeting restrictions R.1 and R.2, known as *phrase-structure grammars*, are not adequate for the formulation of a full grammar for natural language. In place of the grammars in the hierarchy, Chomsky proposed *transformational grammar* as a framework for the description of natural language. A transformational grammar is a rewriting system whose rules can be partitioned into two sets: a *base*, which is a Type-1 phrase structure grammar, and a *transformational component*, which is a collection of tree-transformation rules. The two sets of rules are ordered, so that rules of the base apply first, to derive "deep" syntactic trees. Rules of the transformational component then apply cyclically to the trees generated by the base, to derive "surface" trees, from which a terminal string may be derived. The language generated by a transformational grammar is the set of all terminal strings that surface trees yield. The generative power of transformational grammar is equivalent to that of Turing machines (Kimball, 1967; Peters and Ritchie, 1973). Although transformations play a reduced role in the Government-binding model, this generative power is probably also available in GB grammars. A detailed account of the Government-binding theory is given in Chapter 2.

Since the time the Chomsky hierarchy and transformational grammar were introduced, a number of other grammatical models have been proposed for the description of languages whose properties cannot be conveniently described by phrase structure grammar alone. Two-level grammars (van Wijngaarden, 1974) and affix grammars (Koster, 1971) have been used for the definition of programming languages. Attribute grammars, introduced by Knuth (1968), are powerful tools for the definition of the syntax and semantics of context-free languages. Although the generative power of the newer formalisms is equivalent to that of Turing machines,

they are of significant linguistic interest in the sense used by (Chomsky, 1959). Suitable structural information may be obtained for each string generated by these grammars. In the case of attribute grammars, semantic and context-sensitive properties of strings may be captured directly on the syntactic structure. A formal definition of attribute grammar is given in Chapter 3. Two-level and affix grammars are not considered further in this thesis.

A question that immediately arises in connection with generative systems is the manner in which they may be put to use in *analysis* or recognition of strings over the vocabularies they assume. The problem is highly non-trivial, as attested by the amount of attention it has received in the design and implementation of programming languages. The recognition problem concerns the mapping from a generative grammar satisfying certain restrictions to an automaton which accepts exactly the set of sentences in the language defined by the grammar. The correspondence between grammatical families on the Chomsky hierarchy and automata has been established, and effective procedures exist for converting from one to the other (Hopcroft and Ullman, 1979).

Unfortunately, a similar result does not exist for transformational grammar. The analysis procedures reported in the literature for special classes of transformational grammar are analysis-by-synthesis procedures (Matthews, 1962; Petrick, 1965; Sharp, 1985). While attribute grammar has Type-0 generative power, and hence is also undecidable in general, a substantial theory of parsing with attribute grammars exists. For certain special and well-identified classes of attribute grammar, there exist programs called parser-generators which take a statement of the grammar and automatically produce an efficient program that recognizes all the sentences of the language defined by the grammar. The problem of deriving analysis procedures for languages defined by an attribute grammar is much more manageable and understood, due to the simpler and uniform model of attribute grammar.

Although the technical use of the term "generate" in generative grammar is neutral in considering language from the point of view of generation or recognition, it remains a fact that a separate theory of linguistic performance, drawing on the abstract theory of competence, should explain the manner in which language is put to use. The theory of performance should establish the connection of particular competence grammars to, eventually, analysis and generation automata. The theory of performance should explain the structure of the automata assumed, provide a procedure to map from particular grammars to automata and, where appropriate, state particular limitations that may exist on the automata, such as memory organization and bounds.

Performance plays a role as important as learnability in deciding the explanatory status of a linguistic theory. The empirical question remains about the manner in which the generative devices assumed for linguistic description may be systematically related to analysis ones.

1.3 Grammatical symbols in language description

The rewriting systems discussed above, including early transformational grammar, operate on atomic symbols drawn from a finite vocabulary.[3] In more recent linguistic theory, however, the grammatical symbols of a language are assumed to be more elaborate. Chomsky (1965) defines the so-called Standard Theory of transformational grammar and extends the rewriting system, so that a subset of the rules operate on complex or attributed symbols. A *complex symbol* is an arbitrary collection of attribute-value pairs. Some of these attributes can encode, for example, the categorial information of the symbol. A more complex notion of complex symbol, in which attribute values can be recursively defined, is used in the feature-based formalisms discussed by (Shieber, 1986). In attribute grammar, the syntactic categories assumed are atomic, while each category is formally associated with a fixed set of attribute fields. The rewriting rules are stated in terms of atomic symbols, but the attribute fields of each symbol can be used to state conditions on the application of rules. By a simple notational convention, context-free rewriting in terms of complex symbols can be shown to be subsumed by attribute grammar.

The extension of grammatical symbols from atomic to attributed or complex increases in a non-trivial way the descriptive and generative power of a grammar. In attribute grammar, the extension, coupled with the attribution functions used to define attribute values, increases the generative power to that of Turing machines. This is proved in Chapter 3. Complex symbols are used in current linguistic theories, including Government-binding, Lexical Functional Grammar (LFG) (Bresnan, 1982), and Generalized Phrase Structure Grammar (GPSG) (Gazdar, Klein, Pullum and Sag, 1985). LFG uses a context-free grammar augmented with several notational conventions to define a class of representations, functional structure, which may be viewed as recursive complex symbols. Attribute values in the complex symbols are established by the annotations on the context-free rules. Without any further provisos, the generative power

---

[3] The notion "atomic" in "atomic symbol" is used with its Greek sense; i.e., *indivisible*. An atomic symbol is distinguished exclusively by its name and is not further analyzable. In contrast, the "complex symbols" introduced below have internal structure

of this system can be shown to be Type-0 (Pereira and Warren, 1983). LFG constrains this power by imposing a global constraint on the rewriting base, the off-line parsability constraint, which requires that the degree of ambiguity of the system, for each string, be bounded by a computable function of the string length (Bresnan and Kaplan, 1982).

Generalized Phrase Structure Grammar (Gazdar, Klein, Pullum and Sag, 1985) is a two-level system, which uses feature structures on which rewriting rules and metarules operate. GPSG factors the definition of rewriting rules into two sets of statements, immediate-dominance and linear-precedence statements. Metarules are stated in this format also. As in LFG, several feature conventions and defaults are used to define attribute values. The feature notation and metarules capture generalizations in the grammar and permit considerable gains to be made in succinctness of the grammar, when the size of the GPSG grammar is compared with that of an equivalent context-free grammar. The weak generative power of the system is context-free, although the recognition complexity has been shown to be exponential (NP-hard) in grammar size (Ristad, 1986).

Government-binding theory (Chomsky, 1981) falls within the framework of the Extended Standard Theory, presented in (Chomsky, 1970, 1977). As such, the notion of complex symbol is part of the theory, and a subset of the rules in the phrase structure component, those for lexical insertion, are explicitly defined to operate on complex symbols. Other rewriting rules, including transformations, operate essentially on atomic symbols.[4] However, Government-binding theory contains several other components which play a crucial role. These components are variously known as *modules* or subtheories and define (i) relations between nodes of the syntactic representations produced by the phrase structure component, (ii) markings or annotations on the same representations, and (iii) various well-formedness conditions on the representations. Examples of relations between nodes in representations include *government* and *binding*, from which the name of the theory is derived and which play a prominent role throughout. Examples of annotations on representations include grammatical Case, thematic roles, and various sorts of indices. The well-formedness conditions may be split into two classes, those which constrain phrase structure configurations, and those which constrain occurrences of certain kind of annotations. Conditions of the first kind include the Subjacency condition, while those of the

---

[4] The qualification "essentially" is needed since, adopting the X-Bar theory of (Chomsky, 1970), categorial information of grammar symbols is encoded in categorial attributes. Rules in the syntactic component have access to this information.

second include the Case filter and thematic criterion. All these elements of the theory will be presented in Chapter 2.

While phrase structure rules, except those of lexical insertion, are defined essentially on atomic symbols, it is clear that the subtheories in Government-binding theory refer to complex symbols. We may capture all annotations on syntactic representations defined by the modules as attributes of complex symbols. We will argue in Chapter 2 that all grammatical symbols in the theory are in fact attributed symbols. Thus, annotations such as grammatical Case, thematic role, and referential and chain indices will be defined formally as attributes on categories in the grammar. The modules in the theory define the way in which these annotations are to be evaluated and the conditions that exist on them.

The statements made by the subtheories in Government-binding refer, in general, to an entire syntactic representation. Thus, for instance, the subjacency condition requires that two nodes which have been coindexed by application of the rule move-α must not be too far apart, in a sense which may be made precise by reference to paths in trees. The statement applies to the entire set of nodes in a tree. Because of this feature of the statements in the theory, they may not be directly associated with any one rewrite rule in the grammar. In this respect, Government-binding is significantly different from the two theories discussed above, LFG and GPSG. With the adoption of attribute grammar as a formalism for the statement of the theory (at least of the performance component), we will reduce all statements in the theory to highly local statements, referring to attribute fields in one elementary tree. An *elementary tree* in a syntactic representation is one subtree consisting of the parent node and all its immediate descendents (children). This refinement of the theory, from statements over global tree domains to local ones, permits all statements to be associated with specific productions in the phrase-structure component, precisely in the manner of attribute grammar.

Government-binding theory requires a grammatical framework in which the relevant linguistic unit is the complex or attributed symbol. This framework has basic similarities with attribute grammar, although the two frameworks diverge in two important respects. First, unlike attribute grammar, current Government-binding grammar is a transformational theory. More importantly, given the reduced role of transformations, the statements made by Government-binding theory apply to entire syntactic representations and hence may not be directly captured in an attribute grammar. Given that attribute grammars are already rather general devices, it is of interest to know if the substantive elements of Government-binding may

be reduced to an attribute grammar. In Chapter 3 we argue that this is the case, by reducing the main transformational operation in the theory, rule move-α, to an interpretive mechanism, the Chain rule, defined purely by local attribution statements.

1.4 Substantive and formal properties of language

The statements that linguistic theory makes about the properties of natural language may be of two kinds. On one hand, the statements may be of a specific nature, concerning the particular grammatical symbols, features, and rules that occur in the description of a particular language or class of languages. Statements of this sort are called *substantive*. On the other hand, statements about of the *class* of symbols and rules that may be used in the description, or properties of them, ignoring the particular ones that may be involved, are called *formal*. Chomsky (1965) introduced this distinction between the two kinds of statements that a theory may make.

An important element of a linguistic theory is a declaration of the *grammatical formalism* or language within which the statements of the theory are made. This language should be precise and have sufficient expressive power to permit the statements of the theory to be made. In the limit, the linguistic theory is completely defined by the formalism it chooses. However, in realistic cases this does not happen. Linguistic theory and grammatical formalisms do not in general coincide, and this gives rise to the distinction between substantive and formal properties of language. Formal properties are those which may be shown to follow from the formalism alone. Substantive properties require additional reference to the particular statements made by the theory.

To illustrate the new notions, the formalism within which the Government-binding theory is stated has not been fully characterized in the literature. However, given that GB theory is within the framework of Extended Standard Theory, this formalism should be a revised version of the formalism of transformational grammar, for example, as defined by Peters and Ritchie (1973). Thus, the postulation of transformations and the conditions associated with them, such as recoverability of deletion, are claims about the formal properties of natural language. Similarly, the properties of base grammars defined by the X-Bar theory, as introduced in Chapter 2 and (Kornai, 1983), are formal properties of grammar as well. On the other hand, the particular base rules or transformations that may be written within the framework of transformational grammar

and which Government-binding assumes are not formal elements of the theory, but rather substantive ones.

In this research we consider attribute grammar as a formalism for the statement of linguistic theory. According to this choice, we will argue for an interpretive version of the Government-binding theory. We will show that attribute grammar is sufficient and adequate for the statement this version of the theory. Two formal claims that may be made are (i) that all categories of the grammar are attributed symbols, and (ii) that the non-transformational framework of attribute grammar is sufficient for the description of natural language.

These claims, by themselves, are somewhat trivial given the Type-0 generative power associated with attribute grammar. It is important to associate a substantive statement with the previous two. We develop a specific attribute grammar fragment that implements each of the subtheories of Government-binding, except for Control. In particular, we propose an attribute mechanism, the *Chain rule* mentioned in the previous section, to establish trace-antecedent relations in phrase structure trees. The Chain rule is a set of attribution statements on elementary trees. The Chain rule assumes the functional typology of empty categories established in the GB framework, and the subtheories of Government, Case, and thematic relations. We believe the status of the Chain rule as a universal rule should be identical to that of the noted elements in GB theory.

In this work we do not consider possible improvements of the attribute grammar formalism, including notational conventions. The focus is to capture the substantive elements of the Government-binding theory in the formalism assumed.

1.5 The relation between Government-binding and attribute grammar

It will become apparent as we proceed that the attribute grammar formalization developed here differs form the original Government-binding theory in three respects: (i) the assumption of a transformationless model, and hence of a new set of *levels of representation*, (ii) what might seem to be excessive reliance of phrase structure to define grammatical relations and processes (such as Government, Case, and thematic role assignment), and (iii) the emphasis on computationally

implementable, rather than axiomatic specifications of the grammatical processes assumed in the theory.[5]

Attribute grammar is a formalism different from those previously used for the study of natural language, although, as noted earlier, it is related to the feature-based formalisms discussed by (Shieber, 1986). The notion of *level of representation* has not been used previously in connection with attribute grammars or in formal language theory. In Chapter 2 we define that notion in the context of attributed definitions, although we do not attach formal significance to it. The notion is introduced mainly to help establish the connection between our specification and Government-binding.

With these remarks in mind, the first point can be answered rather directly and clearly. We assume, as seems natural, that a transformationless model of grammar is compatible with what we might call an instantiation, or at least a variant of GB theory. Such transformationless models have in fact been entertained in the past in connection with Government-binding, by Koster (1978), Chomsky (1981), and Barss (1983). The transformationless model developed here does not have a level of D-Structure, although it has an S-Structure level. S-Structure contains all information present at D-Structure in the standard GB model, and hence a D-Structure representation can be derived if a precise correspondence is desired. This derivation, however, would be a useless exercise, given that D-Structure is not an interface to the mental system of language, in the manner that logical form and phonetic form are.

The second point, excessive reliance on phrase structure, arises because of the different formalisms used for the statement of competence and performance grammar. As noted earlier, the major component of GB theory is a set of modules or subtheories whose statements are "global," in the sense that they may refer to entire syntactic representations. In contrast, attributed definitions are stated in a syntax-directed manner, with explicit reference to phrase structure rules. Attribution statements in attribute grammar are given on a production by production basis. The attribution is strictly local, as in any statement access is limited to the attribute fields of symbols in one production. This feature of attribute grammar has important advantages, particularly for compiler writing (Waite and Goos, 1984; Watt and Madsen, 1983).

---

[5] The three points noted capture the criticism of G. K. Pullum to (Correa, 1987a) after its presentation. The criticism has to do with the relation of the attribute grammar given to the Government-binding theory. The remarks in this section attempt to answer this question.

The choice of attribute grammar as a language for formalization of Government-binding theory requires us to decompose the statements of the theory into collections of more primitive (but equivalent) statements, which may be associated with individual phrase-structure rules. This observation is abstract enough to warrant an example for illustration. Let us consider the phenomenon known as Exceptional Case Marking in GB theory. It will be addressed in detail in Chapters 2 and 3. GB theory allows that certain verbs, such as "believe" and "want," may be lexically marked by a feature, so that they may assign grammatical Case to the Subject of their clausal complement. This statement of the theory thus relates two non-adjacent nodes in a syntactic representation: a verbal node V and a noun phrase node NP, in the clausal complement of V. The nodes V and NP are generated by different productions in the grammar. To capture this statement in the formalism of attribute grammar, it must be decomposed into a collection of more primitive statements, each of which is associated with an individual context-free production. The exceptional Case assignment is done through intermediate nodes on the path between the target nodes V and NP. The process is formulated in detail in Chapter 3.

Decomposition of the statements of the theory into simpler ones constitues what (Chomsky, 1981) calls "details of realization;" they clearly depend on the formalization language chosen, in this case attribute grammar. The decomposition is certain to include details not found in the original formulation. On the other hand, it is not clear that psycholinguistic observations will be sufficient to justify all details. The details may be motivated, in part, by other considerations such as adapting the grammar to a given parsing style. Given the intricacy and level of detail of attribution statements, it is important to show that some suitably parameterized version of them is universal, or common across languages. This permits their statement to be preserved across languages. This has not been done in this dissertation, given that we have considered only English. But we hope this demonstration is feasible, in fact, clear to experienced syntacticians.

The context-free grammar underlying the attribute grammar may be interpreted as the grammar which arises from the X-Bar theory assumed in Government-binding, with language-specific parameters for English. We have resorted to encode categorial information in distinct category labels, rather than by means of categorial features, as in (Chomsky, 1970). Given that the set of categorial features and their domains (usually Boolean) are finite, this represents only a matter of execution. The use of distinct category labels permits direct application of "tabular" parsing methods. We have made some effort in Chapter 2 to make explicit the relation between the context-free grammar assumed and the X-Bar grammars commonly invoked in linguistic theory.

The formulation of parameterized X-Bar grammars should benefit from consideration of explicit context-free production sets of the sort developed here.

With regard to the third point, GB theory is, as noted earlier, an abstract specification of the linguistic competence of ideal native speakers. It presents axiomatic definitions of linguistic structures at various levels of representation. The method of axiomatic definition, also known as "principle-based approach," uses extremely general rewrite and annotation rules for the generation of syntactic representations. The bulk of the grammar is in modules or subtheories which constrain the class of representations that may be produced.

The manner in which the theory of competence is related to analysis and generation devices is addressed by the theory of performance. Investigations of performance may advance only as far as understanding of competence permits (Chomsky, 1965). Given the high level of abstractness of the theory of competence and the necessarily low-level nature of perceptual devices, it is unlikely that the relation between the two may be established directly, in one step as it were. It will not be surprising to find that the relation between the two is established through several intermediate levels of abstraction. In this research we explore the use of one intermediate level, which we call "performance grammar." A performance grammar is a generative grammar, in the sense used by the theory of competence, but in contrast to the theory of competence, it assumes a simpler set of devices which may be related more readily to perceptual mechanisms. Our formalism for the study of performance is attribute grammar, as noted earlier.

Part of the claim associated with a theory of competence is that native speakers compute and use the various structures it describes (or isomorphic forms, under some notion of isomorphism). But we do not assume that native speakers use, in any sense of this word, the devices postulated by the theory to generate the linguistic structures. For example, we do not assume that transformational rules need apply in the derivation of S-Structure, or that a free-indexing rule is part of the psychological model of binding -- it may, but this is an empirical issue. Nor do we assume that all levels of representation must be explicitly reproduced during analysis or generation of sentences by the mechanism that carries out these processes. This will be seen clearly here and is discussed in (Johnson, 1987).

While we run the risk of making irrelevant certain formal claims that a linguistic theory may make, the substantive facts of language should remain constant across the theories of competence and performance. In this research we require correspondence at certain levels of

representation between an interpretive theory of performance and a (possibly) transformational theory of competence.[6] The levels of representation at which correspondence is required may be taken to be of interest for some reason beyond the theory. For example, we might be interested in the interface of linguistic faculties to other mental systems. Here we assume this level is S-Structure. We note that all other levels in the standard GB theory, D-Structure, phonetic form, and logical form, may be derived readily from S-Structure.

The claim that the attribute grammar defined here is an instantiation of the Government-binding theory is a difficult one to substantiate, partly due to fuzziness of what might be meant by "instantiate," and also partly due to the abstractness and degree of explicitness of GB theory, as currently available. Government-binding is a theory of competence. Readers who do not agree with the "instantiation" claim may simply ignore it (although the attribute grammar was motivated and made possible by the theory). The results presented here do not depend on this claim, and the attribute grammar for English that we develop may be taken for its own merits. Attribute grammar presents a new way of describing and studying linguistic phenomena. It has a definite advantage over other grammatical formalisms, since it permits systematic derivation of analysis procedures for the language it defines and formal study of its properties.

1.6 Goals and rationale of the dissertation

This dissertation presents a computationally oriented formulation of the Government-binding theory outlined by Chomsky (1981) and describes a method for the derivation of a practical analysis procedure for English. The instance of Government-binding presented here is defined by an attribute grammar, a well-developed and understood grammatical formalism commonly used for the specification of programming languages and syntax-directed translations. The analysis procedure has been implemented in a Prolog-based parsing program.

This enterprise is of theoretical interest for two reasons. First, it will provide an account of linguistic principles within a much simpler grammatical formalism than transformational

---

[6] We believe that if the study of performance succeeds while restricted to purely interpretive (non-transformational) devices, it will be extremely difficult to justify a transformational approach to the study of competence. In this sense, the research presented here argues for a fully interpretive theory of grammar.

grammar. Second, the enterprise is a serious attempt to develop a theory of performance for the Government-binding model.

In the attribute grammar formalism, attributed definitions capture the substantive aspects of Government-binding. In particular, the attribute grammar defines interpretively all relations between nodes of derivation trees which are transformationally defined in the current GB model. Given the reduced role of transformations in Government-binding theory, compared with that of the earlier Standard Theory, the attribute grammar formalization of Government-binding may turn out to be empirically undistinguishable from the transformational version (Chomsky, 1981).[7] An extended argument for a non-transformational theory of grammar is found in Koster (1978). Koster argues that an interpretive theory with conditions only on syntactic representations is of major interest for the simplification it brings to the theoretical framework. Barss (1983) argues for an interpretive approach to chain formation, on grounds of empirical adequacy. Chain formation is the most important motivation and role of transformations in the current GB model.

By presenting a computational instantiation of the theory, we provide an answer for the manner in which knowledge of language is put to use. This reduces the gap that has existed thus far between the theories of competence and performance in the GB model. Attribute grammar is a formal device with sufficient internal structure, for which several methods are available to transform it into equivalent automata. To obtain our analysis procedure in Chapter 4 we apply one of such methods.

Some comments are in order regarding relation of this work to other studies of performance in the Government-binding framework, by (Berwick and Weinberg, 1984) and (Sharp, 1985). Berwick and Weinberg (1984) provide a computational model of Government-binding grammar, assuming the parsing automaton of (Marcus, 1980). This automaton maintains two data structures, a push-down stack and a small constituent buffer. The control of the automaton consists of pattern/action rules, whose patterns match against the constituents in the buffer and

---

[7] This is clear, in a weak sense, if we accept that both attribute grammar and Government-binding have equal generative power, that is Type-0. See also (Chomsky, 1981, p.89-92).

push-down stack, and whose actions perform operations on the two data structures.[8] The computational model of Berwick and Weinberg is obtained by a direct mapping from the transformational theory of competence to the control instructions that the automaton accepts. The Marcus parser implements a transformational GB theory in the sense that for every input string it computes an annotated surface structure, as defined by the theory.

The attribute grammar formulation of Government-binding is an attempt to develop the theory of performance along somewhat different lines. Attribute grammar is used as a specification level intermediate in the mapping from the abstract theory of competence to the particular automata that may be used for parsing or generation. Attribute grammar is an generative device neutral regarding the choice of parsing automata. Thus, attribute grammar affords development of models of performance from the abstract specification of competence by stepwise refinement. Grammatical rules and principles may be stated in a declarative fashion, although in a language with closer connections to performance models. Concrete problems of realization may be formulated and studied in the attribute grammar framework, without worrying about the procedural interpretation associated with the grammar.

Sharp (1985) develops a rather comprehensive analysis procedure for Government-binding. Sharp assumes a transformational version of Government-binding, and defines certain procedures for computation of the "inverse" of the transformations assumed during parsing. The procedure implemented is an analysis-by-synthesis procedure. The work related here differs from his in the use of attribute grammar for a separate statement of the grammar used in the program and, more significantly, the use of an interpretive model in that grammar. Also, the theoretical account of Government-binding given in Chapter 2 is more exhaustive than that provided by Sharp. This permits improvement of the efficiency with which the principles arrived at are used in the attribute grammar. We note the functional classification of empty categories developed in Chapter 2, which permits determination of the type of a category without reference to its antecedent. Similarly, the binding rule developed in Chapter 3 is a refinement of the axiomatic definition of the binding theory. Hence, the new binding rule is more directed and psychologically plausible as a model for the resolution of pronoun and anaphor reference.

---

[8] Berwick and Weinberg (1984, p.153) remark that the Marcus parser is "an informal machine version of an LR(k) parser."

The attribute grammar we develop is for English. We expect that similar specifications to the one developed here can be made for other natural languages. This will provide a basis for the study of the mechanisms underlying natural language. We consider that the most important contribution of this thesis is the disassociation of the notions of "competence grammar" and "performance grammar" in linguistic theory and the instantiation of the system of principles and parameters in Government-binding theory into a precise computational framework, to which some psychological reality can be attached. The attribute computations specified in the attribute grammar are of a more detailed and concrete nature than the abstract principles of GB they purportedly instantiate. We believe that the use of attribute grammars as tools for natural language description and the method followed here to instantiate the rules and principles of the theory are of general interest. As the reader will discover, at each step in the derivation of the grammar reference is made to the original GB theory.

1.7 Outline of the dissertation

The dissertation is organized into five main chapters. In Chapter 2 we develop a detailed account of the Government-binding theory on which this work is based. In Chapter 3 we present an attribute grammar specification of the theory. The main linguistic result of the chapter is the formulation of a Chain rule which applies at S-Structure to establish trace-antecedent relations. In Chapter 4 we introduce the methods by which the attribute grammar specification may be transformed into a parsing program and attribute evaluator, and develop a particular analysis program for English. Discussion of the results and conclusions are given in Chapter 5.

Chapter 2. Government-binding Theory

Government-binding (GB) theory is a grammatical theory that has emerged from work within the transformational-generative grammar tradition. As such, its origin may be traced to the development of transformational grammar in the mid-fifties (Chomsky 1955, 1957) and subsequent reformulations of it, commonly referred to as the Standard Theory (Chomsky, 1965), and the Extended Standard Theory (Chomsky 1970, 1977).

The main statement of the GB theory may be found in (Chomsky, 1981). Since the time of the initial formulation, the theory has gone through a number of major revisions; even the original formulation (Chomsky, 1981) is not a unified theory, but rather a system of leading ideas. At the present time, there are a number of outstanding and competing partial proposals regarding most of the components of the theory, and due to this fact it is extremely difficult to choose among the alternatives proposed to determine what *is* the "Government-binding theory." It is relatively easy to recognize the "Government-binding framework," but I would argue that there is no characterization of it, on which there would be agreement by the parties interested. Even the most fundamental notions of phrase-structure are topics on which there is wide disagreement (cf. Stowell, 1981; Hale, 1983; Kayne, 1984; Rizzi, 1986; Abney, 1987; Williams, 1984; Kiss, 1987).

Given this state of affairs, the objective of the present chapter is simply to present the major concepts and principles of the GB theory, as generally understood, and especially as they relate to this thesis. A significant feature of the chapter is the emphasis it places on the mechanisms of Government-binding grammar: the class of devices it employs for language description, and the particular set of grammatical rules and principles that it proposes to describe human languages, English in particular. Our treatment of these topics, however, is not formal, but only a step in the direction of formalization. There is little by way of examples and motivation of the

principles of grammar proposed. The reader is referred to van Riemsdijk and Williams (1986) for a different outlook and a historical overview of the theory.

2.1 Phrase Structure

In the Standard Theory (Chomsky, 1965), phrase structure arises out of the interaction of two components of grammar, called the *base* and the *transformational* components. These two components make extensive use of a rich set of formal grammatical devices, as will be apparent below. We discuss the phrase structure components in this section. In Chapter 3 we propose an implementation of the theory that reduces the set of formal devices assumed to a minimum, namely attribute grammar.

2.1.1 The Base component

Let V be a finite vocabulary of grammatical symbols. As a temporary approximation, let them be atomic and unattributed symbols. The *base* component of Government-binding grammar is a set of context-free rewriting rules over symbols in V, each satisfying the format (1).

(1) $\quad X \rightarrow \alpha$

In (1), the left-hand side (LHS) symbol $X \in V$ is rewritten as the possibly empty sequence of symbols $\alpha \in V^*$. Each symbol that appears on the LHS of at least one rule is called a non-terminal or *syntactic category* of the grammar. Symbols which appear only on the right-hand side (RHS) of rules are called terminals or *grammatical formatives.* In contrast to other work (e.g., Stowell, 1981), we allow α to be the empty string 'ϵ' .[1] This permits rewriting the symbol X as the empty string, generating the phrase-marker $[_X \in]$. We call such a phrase-marker an *empty category* of category X. The vocabulary of symbols used in the grammar, as well as the context-free form (1) of base rules, are subject to the restrictions of the so-called X-Bar theory, discussed in detail in section 2.3.

---

[1] As noted in Chapter 1, we use the symbol 'ϵ' to denote either the empty string or set-membership. The context in which the symbol is used is sufficient to determine the appropriate meaning.

In order to simplify the statement of transformational rules, we assume that base rules introduce brackets surrounding the string generated, as in (2). With this convention, the bracketed strings generated by the base are phrase-markers.

(2) $X \rightarrow [_X \alpha]$

The bracketing introduced in (2) simply encodes the syntactic structure of the strings generated, and is an extension of the base which does not increase its context-free generative power. The brackets introduced are new terminal symbols, outside the initial vocabulary V of the grammar. A debracketization function may be defined to compute the *yield* of a bracketed string by deleting all brackets in it. It is convenient to let $V' = V \cup \{ [_X : X \in N \} \cup \{] \}$ denote the vocabulary of the extended base grammar.[2]

2.1.2 Lexical insertion

Base rules have traditionally been partitioned into two classes, according to whether they introduce a single grammatical formative (Chomsky, 1965). Rules which introduce a single formative, as in (3), are called *lexical rules*, and distinguished from other base rules. The distinction arises due to the assumed nature of grammatical formatives and the possibility of specifying context-sensitive insertion conditions. A grammatical formative is a "complex attributed symbol," as elaborated below, rather than an atomic entity.[3]

(3) a. $N \rightarrow$ *John*

b. $V \rightarrow$ *break*

c. $P \rightarrow$ *in*

---

[2] Bracketing grammars appear in the earliest formalizations of transformational grammar (Petrick, 1965; Peters and Ritchie, 1973). Given a CFG $G = (N, T, P, Z)$, a bracketing grammar G' corresponding to G may be obtained by letting $G' = (N, T', P', Z)$, where $T' = T \cup \{ [_X : X \in N \} \cup \{] \}$, and $P' = \{ X \rightarrow [_X \alpha] : X \rightarrow \alpha \in P \}$ (Petrick, 1965).

[3] The distinction between lexical and non-lexical categories in (Chomsky, 1965) is subtle. N and V are lexical, but Det is not. The status of modals (M) is not clear.

Abstracting away from the particular symbols involved in (3), we will say that categories that appear on the left-hand side of those rules are *lexical*, unless they are declared ahead of time as *non-lexical* categories. The extension of lexical categories is listed in a *lexicon*, while that of non-lexical is not. Categories which do not introduce formatives will be called phrasal or *major* categories. The convention we make is purely terminological and leads to a partition of the non-terminal vocabulary of the grammar into lexical, non-lexical, and major categories.

The principal motivation given by Chomsky (1965, p. 79) to distinguish lexical from other categories (non-lexical and major) is the need to include *subcategorization* information in lexical categories, but not in the other categories. Subcategorization concerns distinctions that must be made among members of a syntactic category's extension. For example, nouns need be distinguished according to whether they are *count* (as opposed to *mass*), *common* (as opposed to *proper*), or *human* (as opposed to non-*human*). Similarly, verbs need be distinguished according to whether they are transitive, whether they may take a clausal complement, whether they require a *human* Subject, or whether they may assign grammatical Case to the Subject of their clausal complement, under special circumstances.

In early attempts to formalize generative grammars (Chomsky, 1955, 1957), subcategorization was dealt with by context-sensitive rewriting rules, over a domain of atomic category symbols. However, referring to work by G. H. Matthews, Chomsky (1965) points out that rewriting rules are not the appropriate device to effect subcategorization. Subcategorization information is typically non-hierarchical and often involves cross-classification, as suggested by the examples above. Among the distinctions made in subcategorization, typically none has dominance over another. Subcategorization information must instead be represented by sets of *syntactic features* or attributes, which often turn out to be binary-valued.[4] The notion of linguistic feature was introduced earlier by the school of structural linguistics and was adopted in (Chomsky, 1965) as part of the mechanism of transformational grammar. The matrix of subcategorization features associated with a formative is called a *complex symbol.*

We have seen that rewriting rules on atomic category symbols are not appropriate to generate lexical symbols with the necessary subcategorization information. The general problem of

---

[4] The term *feature* is used ambiguously to refer either to an ordered pair consisting of a feature name, such as *pronominal*, and its value, such as '+', or to just the feature name. Also, the terms *feature* and *attribute* are used interchangeably.

extending the expressive power of phrase structure grammar to permit statement of subcategorization information in the rewriting rules was left open in (Chomsky, 1965). Chomsky proposes instead to revise the format of rules like (3), so that symbols representing lexical categories are rewritten into complex symbols (feature bundles) as in (4.a). The complex symbol Q = [+X,±*common*,±*count*,...] generated by this rule may be replaced by a grammatical formative *t* according to the *lexical insertion condition* (4.b). The lexicon is assumed to be a set of pairs of the form $<t, C>$, where *t* is a grammatical formative and C a collection of specified syntactic features associated with *t*. The use of complex symbols in rewriting rules as in (4) was suggested by similar rules in the study of phonology.

(4) a. $X \rightarrow [+X, \pm count, \pm common, \ldots]$

b. **Lexical insertion condition**:
If Q is a complex symbol of a preterminal string and $<t, C>$ is a lexical entry, where C is not distinct from Q, then Q may be replaced by *t* (Chomsky, 1965).

The lexical insertion rule (4) solves the problem noted for the generation of lexical categories with appropriate subcategorization information. It turns out, however, that subcategorization information must also be represented in "projections" of lexical categories -- i.e., in other non-terminal categories.[5] For example, the syntactic features *anaphoric* and *pronominal*, which encode important subcategorization information of nouns, must be associated with and synthesized by the parent node NP. Verbs subcategorize for noun phrases, not nouns alone. This kind of propagation of syntactic information from lexical categories to parent nodes is typical in Government-binding grammar and most current linguistic formalisms. Hence, a more accurate picture of the grammar shows that *all* categories are complex symbols. The need to distinguish lexical, non-lexical, and major categories in the formalism vanishes (although the distinction might still be a useful terminological one). In Chapter 3 we will see that the more recent attribute grammars (Knuth, 1968) or augmented phrase structure grammars (Heidorn, 1972) provide well-established frameworks for dealing with attributed symbols conveniently and uniformly throughout the grammar.

---

[5] We depart in this from Chomsky's observation that "the only categories involved [in subcategorization] are those containing lexical formatives as members" (Chomsky, 1965; p.79).

We return to the use of complex symbols and the need for subcategorization information in section 2.2. We refer to the rule schema (4) as the *lexical insertion rule.* Its application is constrained by match of the subcategorization features of X and those of the formative *t*, as discussed in section 2.2.

2.1.3 Transformational component

The transformational component operates on the phrase-markers derived by the base and the lexical insertion rule (4). A *transformation* is a map on phrase-structure markers (over V'*), and is defined by a schema for a possibly infinite number of unrestricted rewriting rules. An unrestricted rewriting rule is a rule $\alpha \rightarrow \beta$ for rewriting the non-empty sequence of symbols $\alpha \in$ V'* into the possibly empty sequence β, also in V'*. A transformation allows the use of *syntactic variables* over V'* to describe the left- and right-hand side strings in the rule. A transformation is a schema as in (5) for a collection of unrestricted rewriting rules.[6]

(5) $\alpha_1 X_1 \ldots \alpha_k X_k \rightarrow \beta_1 X_1 \ldots \beta_k X_k$

In (5), the α_i are strings over V', and the X_i are variables over strings in V'*. The LHS of the rule defines acceptable *factorizations* of the input phrase-marker, into substrings α_1, X_1, ..., α_k, X_k, such that when concatenated the original phrase marker is obtained. For each $i = 1, \ldots, k$, α_i is to be rewritten as β_i, and X_i is to be kept or rewritten identically in the output phrase marker. (5) is a schema for the infinite number of unrestricted rewriting rules that result from substitution of each X_i by an actual string over V'. In Standard Theory terminology, the LHS of the transformation is called the *structural description* and the RHS the *structural change.*

In early transformational grammar, transformations are unrestricted operations that can perform arbitrary rewriting of the input phrase-marker. The work of Chomsky (1964), Ross (1967a), and Emonds (1976) developed a general theory of formal constraints on transformations. These constraints took two forms: (i) constraints on the kind of structural descriptions allowed in the formulation of transformations, and (ii) constraints on the kind of rewriting that transformations

---

[6] It is necessary to supplement the informal definition of transformation just given so that the output of a transformation is a phrase-marker; note that not all strings over V'* are phrase-markers. We omit this requirement from our definition for simplicity in the exposition.

can perform. It is of interest to observe, perhaps only for historical reasons, that none of the constraints that were imposed on transformations resulted in any restriction of their generative power (Kimball, 1967; Peters and Ritchie, 1973). This result is analogous to that of formal language theory, by which unrestricted Phrase-Structure, Type-0, Context-Sensitive with Erasing, and Non-terminal Rewriting grammars are all equivalent, in the sense that they generate the same class of languages. These four classes of grammar generate the Type-0 languages, in spite of the formal restrictions imposed on the form of their productions (Harrison, 1978). We do not review here any examples of early transformations or the formal constraints that were slowly imposed on them; the reader is referred to the original sources cited, or to van Riemsdijk and Williams (1986).

2.1.3.1 move-α

It is fair to say that in Government-binding theory only two transformations are assumed: *move-α* and *quantifier-raising*.[7] The rule move-α, defined in (6), has a parameter α which ranges over syntactic categories. In particular, α ranges over the phrasal categories NP, PP, CP, etc., as well as over the inflectional category I (for Subject-Auxiliary inversion).

---

[7] Most formulations of move-α in the current literature define it informally in a more general fashion than we do. It is often assumed that move-α allows either substitution or adjunction of a phrase and that "move-α only optionally creates a trace" (Lasnik and Saito, 1984, p. 254). Under this definition, both our definitions of move-α and quantifier raising are instances of the same operation (I thank Prof. Kornfilt for this observation).

It should be noted, however, that the formalism of transformational grammar, e.g., as in (Peters and Ritchie, 1973), does not have expressive power that permits the statement of move-α proposed by Lasnik and Saito (1984). Following Peters and Ritchie, the mappings effected by transformations are the result of applying three very restricted types of "elementary transformations": deletion, substitution, or (sister or Chomsky) adjunction of the contents of a sequence of terms. The elementary transformations that define a transformation are obligatory rather than optional. Thus, an operation that performs either substitution or adjunction, and leaves optionally a trace, is not a grammatical transformation but a *family* of four distinct, if closely related transformations.

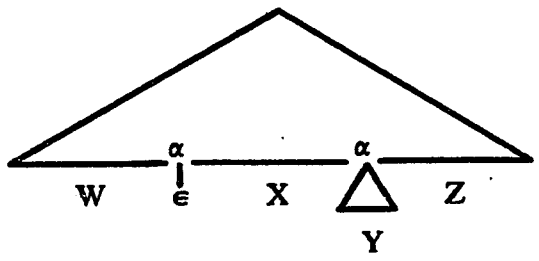
While on this issue, current definitions of move-α in the literature do not specify direction of movement; i.e., movement of a phrase may be either leftwards or rightwards. Again, this option is not possible to state in the older formalism of transformational grammar. The sequence of factors in the structural condition of a transformation is ordered. It is clear that transformations in Government-binding theory are stated in a language with greater expressive power than transformational grammar. It is, in part, this greater expressive power that permits reduction of the class of transformations to one member, move-α.

(6) **move-α:**

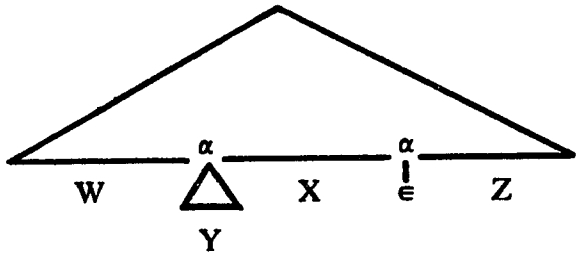
$$W\ [_{\alpha}\ \epsilon\ ]\ X\ [_{\alpha}\ Y\ ]\ Z \rightarrow W\ [_{\alpha}\ Y\ ]\ X\ [_{\alpha}\ \epsilon\ ]\ Z$$

In (6), W, X, Y, and Z are syntactic variables over V'*, 'ϵ' is the empty string, and the two pairs of brackets that appear on each side of the rule are matching brackets -- i.e., the left and right brackets of each pair are introduced by a single application of some base rule, as in (2). The effect of the transformation is shown graphically in (7). The structural description assumes a phrase-marker $[_{\alpha}\ Y]$ of category α, which is moved to the position of the empty category $[_{\alpha}\ \epsilon]$. The two occurences of α need not be at the same depth from the root node of the input phrase-marker. The empty category left by the operation is called a *trace* of the moved phrase.

(7) a. **structural description:**



b. **structural change:**



Two concrete examples of the application of move-α in a derivation appear in (8). We show only the role move-α plays in the two derivations; further details about the derivations are given in subsequent sections. (8.a-b) illustrate the derivation of a passive sentence. (8.a) is a simplified phrase-marker which may be generated by the base; it meets the structural description of (6), with W=Z=ϵ , X="was broken," and Y="the vase" (ignoring brackets). The transformation may apply, yielding the passive surface form (8.b). The second example (8.c-d) illustrates the

derivation of a Wh-question (9.d) from the base-generated form (8.c). The reader may verify that indeed the structural description of (6) is satisfied by (8.c).[8]

(8) a. $[_{NP} \in]$ was broken $[_{NP}$ the vase]

b. $[_{NP}$ the vase] was broken $[_{NP} \in]$

c. $[_{NP} \in]$ [you think Mary likes $[_{NP}$ who]]

d. $[_{NP}$ who] do [you think Mary likes $[_{NP} \in]$]

In (8), α equals NP. The instances of the two applications of move-α could thus be called *move-NP*. However, in Government-binding terminology only the first instance (8.a-b) is called move-NP. The trace left at the extraction site in (8.b) is an *NP-trace*. As we will see in other cases, GB grammar often places greater emphasis on a syntactic feature associated with a node than on the category of the node. In (8.c-d), the phrase moved is a wh-phrase, marked by the binary feature [+ *Wh*]. This fact is of significance in the theory since wh-phrases have the status of (logical) *operators*. The instance of move-α in (8.c-d) is called *move-Wh* and the trace left in (8.d) is a *Wh-trace*. The two types of movement are also characterized by the thematic theory of section 2.7. NP-movement is to argument positions (A-positions) and hence also called *A-movement*; Wh-movement is to non-argument positions (A-Bar positions), so it is *A-Bar movement*.

The move-α transformation (6) is in its bare bones: it defines only the phrase-marker rewriting done by its application. The transformation has an important extension associated with it, the *trace convention*, and several other conditions on its application. These are developed and discussed in the remainder of the chapter.

---

[8] The *do*-support in (8.d) is introduced by the auxiliary system.

2.1.3.2 Quantifier raising

The quantifier raising transformation is often identified as an instance of move-α in the Government-binding literature. However, its structural description and structural change differ from move-α, and hence the identification is not warranted. The mechanism of transformations defined in (Chomsky, 1957) does not provide for the use of disjunction in the statement of transformations, to specify alternative actions to perform. Quantifier raising applies to quantified phrases in phrase-structure representations.[9] As shown in (9), quantifier raising moves and Chomsky adjoins the quantified phrase $[_{\alpha}\ X]$ in the structural description to a position immediately outside the sentential form IP (Inflectional Phrase) in which the phrase occurs.

(9) **Quantifier-Raising:**

$$U\ [_{IP}\ W\ [_{\alpha}\ X]\ Y\ ]\ Z \quad \rightarrow \quad U\ [_{IP}\ [_{\alpha}\ X]\ [_{IP}\ W\ [_{\alpha}\ e]\ Y\ ]\ ]\ Z$$

Quantifier raising is somewhat similar to move-α, but its structural description does not assume a pre-existing empty category as the target of movement. Quantifier raising is not structure preserving, in the sense of Emonds (1976). The Chomsky adjoined structure that results from application of the transformation cannot be generated by application of base phrase-structure rules alone, given the constraints on base rules defined by X-Bar theory. We will have little else to say about quantifier raising and its role in the derivation of logical forms.

2.1.3.3 Empty categories

Move-α (6) leaves an empty category at the extraction site, and assumes a pre-existing empty category as the target of movement. These choices are theory-internal in Government-binding. The choices are motivated by certain central assumptions about the relation of phrase structure to semantic representations, as discussed in section 2.7. Neither of the two choices is logically necessary, for it is equally easy to define an alternate transformation move-α′, as in (10), which (i) leaves no trace at the extraction site, and (ii) does not require a pre-existing empty category

---

[9] A quantified phrase $[_{\alpha}\ X]$ is marked by a binary feature [+Q] associated with the root α of the phrase. The value of Q depends ultimately on lexical features of lexical items contained in the phrase.

as the target for movement.[10] Notice carefully that the output of (10) is string-equivalent[11] to that of (6). The descriptive differences between (10) and (6) are due only to the empty categories assumed in (6), but not in (10).

(10) **move-α′:**

$$W\ X\ [_{\alpha}\ Y\ ]\ Z \rightarrow W\ [_{\alpha}\ Y\ ]\ X\ Z$$

The *structure preservation hypothesis* of Emonds (1970) motivates the occurrence of empty categories in phrase-structure representations and the definition of move-α as in (6). Emonds hypothesis requires that a constituent of category α can only be moved by a substitution rule into another position of category α (Radford, 1981). The position which is target for movement must be present in the input phrase-marker, not adjoined. Emonds' principle is an important formal constraint on transformations. One of its implications, if all transformations obey it, is that any structural configuration that may be transformationally derived, may also be derived by application of base rules alone. The principle, though, is not satisfied by all transformations, as we saw with quantifier raising. In Chapter 3, we make use of the structure preservation property of move-α to develop an interpretive alternative to this rule, leading to a transformation-less model of Government-binding grammar. Further motivation for the use of empty categories comes from thematic and binding theories. They require structural representation of implicit argument positions, inducing structural parallelism between movement structures and relations of bound anaphora (van Riemsdijk and Williams, 1986). Discussion of these topics at this point would lead us far astray; we return to them in the sections below.

The use of empty categories in syntactic representations is one of the major differences between GB theory and other current grammatical theories, including Lexical-Functional Grammar (LFG) (Bresnan, 1982), Generalized Phrase-Structure Grammar (GPSG) (Gazdar, Pullum, Klein, and Sag, 1985), and derivatives. Assuming the framework of augmented phrase structure, discussed in the next section, it is possible to encode the information represented structurally by empty categories in a non-structural way (Jensen, 1988; Gazdar et al., 1985). If ε is an empty category with parent node X, this information can be represented by an attribute SLASH, as in GPSG, on the parent node. We can then let the value of the feature be the bundle of attributes

---

[10] I thank Karen Jensen for this observation. See (Jensen, 1988).

[11] We say that two bracketed strings α and β are *string-equivalent* if their yield is the same terminal string.

associated with ε. Any conditions on the content and distribution of ε may be stated as conditions on occurrences of the feature. At this point it becomes largely a matter of choice (theory-internal) whether empty categories should appear explicitly as nodes in derivation trees.

2.2 Augmented Phrase-Structure

A simplification was made in section 2.1 concerning the nature of symbols in the vocabulary of the grammar. It was assumed that grammatical symbols are atomic and unattributed. This simplification turns out to be extreme and not valid in Government-binding, as we show in this section. We intend to show the prominence of complex/attributed symbols in GB grammar and relevance of attributed definitions in a precise symbolic statement of Government-binding theory. Our notations are informal; we postpone definition of attribute grammar until next chapter.

2.2.1 Complex symbols

The Standard Theory (Chomsky, 1965) introduced the notion of *complex symbol*, as a matrix of specified features or *attributes*. The specified features of a category may be divided into *subcategorization* features of a somewhat semantic nature, and *lexical features* such as ±N and ±V, which can be used to encode the syntactic category of the complex symbol. We first focus our attention on lexical features.

In (Chomsky, 1970), the *lexical categories* N, A, V, and P are analyzed in terms of the binary lexical features ±N and ±V, as in (11). With this convention, the lexical categories may be reduced to a single category-neutral symbol X, with the lexical attributes in (11). Distinctions between categories can now be translated into differences in the lexical feature values associated with X.

(11)

	N	A	V	P
N	+	+	−	−
V	−	+	+	−

The proposal (11) is part of a larger scheme, the X-Bar theory. X-Bar theory proposes that the syntactic categories of a language are "projections" of lexical categories and encoded in terms of the same set of lexical features. The category-neutral symbol X has *projections* X^0, X^1, X^2, and so on; we let $X = X^0$. Each projection X^i of a category, for $i \geq 0$, has the lexical feature analysis

of the zero-level projection X^0.[12] We let *lex*(X) denote the lexical feature analysis of X, and write XP and XB in place of X^2 and X^1, respectively. We return to the X-Bar theory in section 2.3.

The codified categorial information associated with a symbol is important throughout the grammar. X-Bar phrase structure rules are stated in terms of the category-neutral symbol X and its projections and refer heavily to the lexical features associated with the symbols. In (12) we show how the traditional phrase structure rules (12.a-b) may be stated as a single X-Bar rule (12.c), assuming the analysis (11) of the categories involved. In (12.d) we show the equivalent attribute grammar notation that we adopt for the statement of attribute conditions.

(12) a. $V^1 \rightarrow V\ N^2$

b. $V^1 \rightarrow V\ A^2$

c. $X^1[-N, +V] \rightarrow X^0[-N, +V]\ X^2[+N]$

d. $X^1 \rightarrow X^0\ X^2$
condition:
$lex(X^1)=[-N, +V]$
$lex(X^0)=[-N, +V]$
$lex(X^2)=[+N]$

The use of lexical features to classify syntactic categories permits to capture generalizations in the statement of phrase structure rules, leading to reduction in the number of "rules" that need to be written, and hence to potential *simplification* of the grammars that are written. In (12.c-d), the feature specification [+N] permits reference to the class of syntactic categories N and A. The value of the other lexical feature involved, [+V], is left unspecified.

While it is true that the above generalizations lead to reduction in the number of phrase-structure rules that need to be written, the notion of rule in (12.c-d) is quite different

---

[12] The distinctions between "projections" of the symbol X may also be encoded by an additional integer-valued feature *Bar*, whose value is *i* for the *i*-th projection of X. This notation reduces the vocabulary of the grammar to the single symbol X. The production set of the base is reduced, potentially, to the single rule $X \rightarrow X\ X$. The *Bar* feature is adopted in GPSG (Gazdar, Klein, Pullum, and Sag, 1985).

from that in (12.a-b). The extra complexity introduced by the attribution in (2.c-d) might well be an important factor to take into account in whatever simplicity metric proposed. In section 2.3, on X-Bar theory, we ignore the potential for simplification of the base in the manner just described. Since the set of classifying lexical features that might be assumed is finite, and the range of their possible values is also finite, this elaboration of the theory does not increase the formal context-free generative power of the base.

The set of lexical features used to classify syntactic categories is not fully developed in the analysis of Chomsky (1970). It is not indicated, for example, what additional features may be used to classify minor lexical categories, such as adverbs, conjunctions, quantifiers, auxiliaries, etc, if indeed these are to be taken as independent categories. There have been few proposals concerning this problem. Jackendoff (1977) offers the most detailed proposal and explores in detail the manner in which the lexical feature analysis he presents may be used to simplify the base grammar. His system refers to the features [±Obj, ±Subj, ±Det, ±Comp], which are used to analyze ten different lexical categories. Jackendoff's proposal, though, has been severely criticized (Stowell, 1981), and is not in common use.

In addition to the lexical categories above, GB grammar assumes two non-lexical categories I and C, which figure prominently in the grammar. These categories project up to the categories IP for sentence and CP for clause, respectively. The lexical feature analysis that may be given to these categories has not been considered in most X-Bar studies. Stowell (1981) proposes to analyze C as a noun, namely [+N, -V], to account for the parallel distribution of clauses and noun phrases as complements of verbs, and possibly also as Subjects. Chomsky (1981) identifies *AGR*, the main feature set of I, with PRO, an instance of N. C and I may be distinguished from N for other purposes by a feature [±*tense*], present in C and I, but absent in N. Under the X-Bar convention, the lexical feature analysis of a category is synthesized by its projections.

2.2.2 Subcategorization

In the Standard Theory subcategorization features are attributed to grammatical formatives and the lexical categories under which they may be inserted. Examples of these features (Chomsky, 1965) include, for nouns, binary features of a semantic nature such as ±*common*, ±*count*, ±*animate*, ±*human*, ±*abstract*, etc. One role of the features is to permit expression of subcategorization and selectional restrictions that predicates impose on their arguments.

Without entering into details, the semantic oddity of (13) may be attributed to a mismatch between the subcategorization features of the various lexical items in the expression.[13]

(13) Colorless green ideas sleep furiously.

In addition to the noted semantic subcategorization features for nouns, there are features of a more syntactic nature, such as [±*pronominal*] and [±*anaphoric*] for nouns, and the multivalued feature [*subcat*] for verbs. We consider now the last feature. The Standard Theory assumes a subcategorization feature associated with the complex symbols of predicates (mainly V and N). The value of this feature encodes the local syntactic environment in which the predicate symbol may appear. The feature indicates, for example, whether the predicate is transitive, or whether it takes a prepositional or clausal complement. For every base rule (14.a), the subcategorization rule (14.b) applies and assigns the context value *'α_β'* to the *subcat* feature of predicate Y. The notation X.*a*, where X is a category symbol and *a* an attribute name, denotes the attribute occurrence *a* associated with symbol X.

(14) a. $X \rightarrow \alpha Y \beta$, where Y is a lexical category

b. **Subcategorization rule:**
Y.*subcat* ← *'α_β'*

The subcategorization context assigned to Y constrains the class of lexical items that may be inserted under Y. Lexical insertion condition in rule (4) may be made more precise as in (15), so that lexical insertion of *z* under Y (i.e., application of the rule) depends upon match of the *subcat* features of Y and *z* (Stowell, 1981). The value of the feature *z.subcat* is listed in the lexicon, as part of the lexical entry for *z*.

(15) a. **Lexical insertion rule:**
$Y \rightarrow z$

b. **Lexical insertion condition:**
Y.*subcat* = *z.subcat*

---

[13] (13) is the well-known example from (Chomsky, 1957).

2.2.3 Indices

A new class of syntactic features was introduced in the Extended Standard Theory (Chomsky, 1977). These features are collectively referred to as *indices*, and are best seen as integer-valued attributes on nodes in derivation trees.[14] To illustrate the intended use of "indices" in the theory, Chomsky (1977, p. 6) assumes that "each occurrence of a category in deep structure is assigned a numerical index by some general procedure."

There are several instances of indices: the numerical *base-index* proposed above; *trace indices*, which result from the application of transformations; *referential indices*, relevant for the Binding and Control theories; and *agreement indices*, which determine morphological agreement between e.g. a noun phrase and a verb, and are relevant for the definition of the notions of government and binding. We believe it is best to consider indices as a new extension of the notion "complex symbol" in the Standard Theory. In this section we discuss indices introduced transformationally. Other kinds of indices are discussed later in the chapter.

2.2.3.1 Transformational indices and the trace convention

The transformations of section 2.1.3 are mappings on unattributed phrase-markers. That formulation of transformations ignores an important extension, known as the *trace theory* of transformations. This extension is developed primarily in (Wasow, 1972) and (Chomsky, 1977) and is part of the core mechanism of transformations in Government-binding. According to trace theory, application of move-α or quantifier raising to a constituent α has the side-effect of introducing an index which uniquely identifies the moved constituent with the empty position $[_{\alpha}\ \epsilon]$ it leaves behind, after application of the transformation. The exact mechanism may be formulated in several different ways. Chomsky (1977) assumes that the base index independently assigned to the moved category at deep structure is retained by the category and replaces the index of the empty position target of movement; meanwhile, the trace left behind retains the same index.

The proposed extension of the transformational mechanism may be expressed succinctly and precisely as in (16). We assume, adopting Chomsky's proposal, a syntactic feature *Chain*, which

---

[14] Indices are not referred to explicitly as *features* within the EST and Government-binding frameworks (cf. van Riemsdijk and Williams, 1986). Various subscripting and superscripting conventions are used to represent them.

is integer-valued and associated with all nodes at deep structure. Formally, this feature is a member of the complex symbol that labels each node in the deep structure. It differs from other features found at that level, such as subcategorization features, in that its value is not lexically defined. The value of the feature is instead defined at deep structure by Chomsky's "general procedure." For concreteness we may assume it establishes a preorder enumeration of nodes at that level. The value of the feature is also affected by application of a transformational rule. When the transformation applies, the value of the *Chain* feature at the nodes affected in the output tree is defined by the two attribution rules in (16.b). The *Chain* value on nodes in the output tree not affected by the transformation is a copy of the same attribute from the corresponding nodes in the input tree. Attributed rule (16) is our revised form of move-α.

(16) a. **move-α:**

$$W\ [_\alpha \in]\ X\ [_\alpha Y]\ Z \quad \rightarrow \quad W\ [_\alpha Y]\ X\ [_\alpha \in]\ Z$$

b. **Trace convention:**

$\alpha_4.Chain \leftarrow \alpha_2.Chain$

$\alpha_3.Chain \leftarrow \alpha_4.Chain$

The revised rule (16) consists of two parts, a transformational mapping (16.a), and an attribution section (16.b). The transformational mapping is the rule move-α (6), but now operating on attributed phrase-markers. The attribution section coindexes the moved phrase $[_\alpha$ Y] with its trace. Each α_i in (16.b) denotes the *i*-th occurrence of α in the rule (16.a); for example, α_1 is the root of the phrase-marker $[_\alpha \in]$ on the left-hand side.

The trace convention just defined associates a unique trace index *Chain* with each position occupied by a moved phrase in the course of a transformational derivation. At the end of the derivation the moved phrase is uniquely coindexed with all positions it has occupied in the course of the derivation. This is illustrated in (17), where the phrase [*who*] has been moved two times. The subindices attached to the bracketed phrases are instances of the *Chain* feature in the attribution (16.b). A sequence of constituents in a phrase-marker which have been coindexed by application of move-α are said to form a *syntactic chain.* Hence, in (17.c) the sequence ($[\text{who}]_i$, $[\in]_i$, $[\in]_i$) is a chain, namely chain *i.*

(17) a. $[\epsilon]_g$ you think $[\epsilon]_h$ $[\text{who}]_i$ left.

b. $[\epsilon]_g$ you think $[\text{who}]_i$ $[\epsilon]_i$ left.

c. $[\text{who}]_i$ you think $[\epsilon]_i$ $[\epsilon]_i$ left.

There are alternatives to Chomsky's base-indexing proposal to evaluate trace-indices. One alternative permits elimination of the *Chain* attribute from deep structures and evaluation of *Chain* indices only on demand, as needed. We do not develop any of these alternatives here, as the problem is a matter of realization, on which little of theoretical interest hinges.

Anticipating the material Chapter 3, we take there a different view of move-α, as an abstract syntactic relation between nodes in base-generated syntactic representations. This is along the lines of (Chomsky, 1982) (see his (46)), (Koster, 1978), and (Barss, 1983). This offers an alternative to the transformational evaluation of trace indices. Attribution associated with context-free base rules is used to define a *Chain Rule* which establishes trace-antecedent relations in an input phrase-marker interpretively. This rule has, arguably, the same empirical effects as move-α.[15] In this case, there is no strong motivation to prefer move-α over the interpretive Chain Rule, especially in light of Chomsky's remarks (1981) that the main empirical effect of move-α is to establish trace-antecedent relations, and his further remark that the subdivision of the syntax into base-rules and move-α (which is a simplification) is a considerably marginal issue.

We may appreciate the advantages of the Chain rule over move-α if the grammar developed is viewed not just as a model of competence, but also as a model of performance. Several analysis procedures may be defined for the Chain rule proposed. From an explanatory perspective, the Chain Rule leads to simplification of the grammar and improvement of its explanatory status. An entire class of rules, the transformational ones, is eliminated without sacrificing descriptive adequacy. Since the device assumed for the formulation of the Chain rule, attributed definition, is already assumed in the symbolic (as opposed to verbal) statement of the GB theory, the gain achieved is at no extra cost for the grammar. Notice that this "needed on independent grounds" argument is of a more fundamental nature than those advanced in other contexts to justify a

---

[15] Determination of trace-antecedent relations is evaluation of the *Chain* attribute associated with nodes in the input phrase-marker. We say that a rule is interpretive if it does not rely on transformation to achieve its result.

proposal. Typically, an argument would claim that a substantive principle of grammar, such as the ECP or thematic criterion, or a given rule, as move-α, is needed independently, so it is licit to assume the principle or rule as part of one's proposal, at no extra cost for the simplicity metric in the grammar. Our argument is at a different level, since it concerns not particular principles or rules of Universal Grammar (UG), but rather the language (or class of devices) in which UG is to be stated. The Chain rule (or a more refined version of it, perhaps), is a step in the line of research started in the mid-sixties to reduce the role of the transformational mechanism in generative grammar (Ross, 1967a).

The quantifier raising rule defined in (9) also coindexes a trace at the extraction site with the moved phrase, in a manner similar to move-α. The addition of the trace convention to quantifier raising is left as an exercise. We have shown that the base component of Government-binding grammar is a collection of context-sensitive lexical insertion rules and a context-free grammar on a vocabulary of attributed symbols. The transformational component is a collection of tree transformation rules, over a domain of attributed trees.

2.3 X-Bar Theory

X-Bar grammars are rewriting systems widely used in current linguistic theories to define the base of natural language grammars. In this section we review the X-Bar theory, define the vocabulary of symbols that we adopt for GB grammars, and define the production set that we assume for the base of English.

In this discussion we do not trace the historical development of X-Bar theory or the motivations behind it. The reader is referred to (Chomsky, 1970) and (Jackendoff, 1977) for the linguistic motivations. The formal characterization is adapted from (Kornai, 1983) and (Pullum, 1985).

2.3.1 X-Bar grammars

An X-Bar grammar is a context-free grammar G=(N, T, P, Z), with certain structure imposed on its set N of non-terminal symbols, and certain restrictions imposed on the set P of productions. The structure that may be imposed on N and the restrictions that may be imposed on P are independent of one another. We obtain different notions of X-Bar grammar, depending one the choice of non-terminal structure and production restrictions that is made.

2.3.1.1 Structure of Non-terminals

The primary property of an X-Bar system is *lexicality* (Chomsky, 1970; Kornai, 1983; Pullum, 1985). By lexicality, the non-terminal vocabulary of the grammar is a projection of the set of pre-terminal categories assumed, in the sense that we now elaborate.

Let L be the set of pre-terminal categories accepted to describe natural language. This set contains lexical categories such as N, V, P, A, Art, Adv, etc, which correspond to noun, verb, preposition, adjective, article, adverb, and so on, as well as non-lexical categories such as I and C. The principle of lexicality says that the syntactic categories of the language are symbols of the form X^i , where $X \in L$, and $i \geq 0$. For each $X \in L$, i may range up to some positive integer max_X. The non-terminal vocabulary is the set N in (18). Each category X^i is taken as an unanalyzed symbol of length one; the superscript i on the category is to be thought of as part of the print-name of the category, not necessarily as one of its attributes (Kornai, 1983).

(18) $\quad N = \{ X^i : X \in L \text{ and } 0 \leq i \leq max_X \}$

As discussed in section 2.2.1, grammatical symbols may be analyzed in terms of a finite set of binary-valued lexical features. In this case, the principle of lexicality requires, also, that for every pair of categories X^i and X^j, the lexical feature analysis of the two categories be identical -- i.e., $lex(X^i) = lex(X^j)$, for $i, j \geq 0$. If $i \geq j$ $(i = j+1)$ we say that X^i is an *(immediate) projection* of X^j. A category X^i is called an i-bar projection of X^0. We identify X^0 with X, and write X^{max_X} also as XP. Given these conventions, L is the set of zero-level categories.

2.3.1.2 Constraints on productions

The second major principle of X-Bar grammars is *succession* (Kornai, 1983; Pullum, 1985). By succession, the root category in every elementary tree defined by the grammar is an immediate projection of one of the daughters. The daughter category of which the root is a projection is distinguished the *head* of the elementary tree. By extension, if X is the head of Y and Z is the head of X, we say that Z is the head of Y. Eventually, every tree with root X^i is headed by the pre-terminal category X^0, with the same lexical feature analysis as X^i. The formal constraint imposed by succession on the form of X-Bar productions is that they conform to the schema (19), where $i > 0$ and α and β are strings over V' .

(19) $X^i \rightarrow \alpha\ X^{i-1}\ \beta$

The linguistic intuition that the principles of lexicality and succession capture is the traditional notion *head-of-a-phrase*. In natural languages, every phrase of category X is *headed* by a lexical category Y of the same "class" as X. In less symbolic terms, noun-phrases are headed by nouns, verb-phrases by verbs, and so on (Chomsky, 1970). These two principles rule out productions such as NP→ V NP and VP→ N PP from the base of any natural language.

2.3.1.3 Formal definition of X-Bar grammar

There are a number of other X-Bar constraints which may be imposed on the base. Rather than considering them independently, we give them in (20).

(20) Let L be an alphabet. A *strong X-Bar grammar* based on L is a context-free grammar G=(N, T, P, Z), where

(i) (lexicality) $N=\{ X^i : 0 \leq i \leq max_X \text{ and } X \in L \}$
For each $X \in L$ we let $X^0=X$ and $X^{max_X}=XP$.

(ii) (uniformity) $max_X=max_Y=max$, for all $X, Y \in L$

(iii) (centrality) $Z \rightarrow S \in P$, and $S=X^{max}$, for some $X \in L$.
(Z does not occur on the RHS of any production.)

(iv) For each production $X^i \rightarrow \omega \in P$, $0 \leq i \leq max_X$

(a) (succession) $\omega=\alpha X^{i-1}\beta$, if $i > 0$;
$\omega=t$, for some $t \in T$, if $i=0$
We say that X^{i-1} is the *head* of X^i.

(b) (maximality) for $i > 0$, the strings α and β above are over the vocabulary $V_m=\{ X^{max}: X \in L \}$

(c) (optionality) $(X^i \rightarrow \alpha' X^{i-1} \beta') \in P$,
if α' and β' may be obtained by deletion of one or more
symbols from α and β, respectively; $i > 0$.

Following Pullum (1985), we say that a CFG G satisfying the constraints above is an X-Bar grammar. If for all $X \in N$, $max_X = max$, for some integer max, the notion of strong X-Bar grammar defined here coincides with that of strong $X^{(max)}$ grammar of Kornai (1983).

2.3.1.4 Formal properties of X-Bar grammar

The two more interesting results that Kornai (1983) proves about X-Bar grammars are the following.

Lemma 1: A strong X-Bar grammar G may be converted into a (weakly) equivalent one, G', whose vocabulary uses 1-bar level projections only.

This lemma follows directly from Kornai's lemma 1.2.

Lemma 2: The family L_X of languages generated by strong X-Bar grammars is a proper subset of the context-free languages and cannot be compared with the regular languages -- i.e., $L_X \subset L_2$, and neither $L_X \subseteq L_3$, nor $L_3 \subseteq L_X$.

This is Theorem 2.1 of Kornai's.

Since X-Bar grammars are restricted versions of context-free grammars, one might hypothesize that they define a normal form for context-free grammars. Lemma 2 is included here to show that this is not the case. X-Bar grammars are not the normal forms for context-free grammars; the Chomsky Normal Form (Chomsky, 1959) and Greibach Normal Form (Greibach, 1965) are the most common.

2.3.2 Vocabulary of the grammar

We review in this section the non-terminal vocabulary assumed in current Government-binding grammar. As mentioned in the introduction to the chapter, GB theorists are often more interested in the elements of so-called Universal Grammar, rather than in explicit formulations of grammars within Universal Grammar. Insofar as this trend applies to the content of the vocabulary of X-Bar grammar, none of the work within the GB framework of which I am aware, and in particular (Chomsky, 1981), lists explicitly the set of categories assumed. The proposal which is made here is only tentative. It should be remarked that the categories implicitly assumed in GB grammar, as well as the X-Bar conventions about them, have changed since the original statement of the 1981 theory (cf. Chomsky, 1981 and Chomsky, 1986b).

2.3.2.1 Zero-level categories

With these remarks in mind, let us proceed to make the X-Bar vocabulary explicit. The set of *categories* we assume is adapted from Jackendoff (1977) and the set of *category labels* is as proposed by Chomsky (1986b). The lexical categories that we assume are those in (21).

(21) N, A, V, P, Adv, Q, Art, Deg, M, BE1, BE2, HAVE, DO, Cnj

Each category in (21) is a zero-level, pre-terminal category of the grammar, and its extension is listed in the lexicon. Unlike Kornai (1983), we do not assume that the extensions of these categories are disjoint. Brief inspection of a standard dictionary (e.g., Longman, 1978; Webster's, 1963) reveals that most of the lexical categories intersect. The first eight categories, N through Deg, have the standard meanings, as in Jackendoff (1977); the next five categories M through DO partition the class of auxiliaries into modal, progressive, passive, perfect, and "do" markers; the last category, Cnj, corresponds to conjunctions.

We assume two non-lexical categories I and C, defined also as zero-level categories in the grammar. I is an inflectional element, containing tense, aspect, and mood indicators of sentences. These elements may be implicit or are introduced by auxiliary elements. I is the head of the sentence symbol IP. C is the category of complementizers, assumed head of clauses CP, and precedes the sentential symbol IP. The categories I and C do not have a lexical extension listed in the lexicon; hence the motivation for calling them non-lexical. Instead, the main productions that introduce I and C are ϵ-productions, which expand the categories as empty

nodes. There are, however, a few productions that can generate specified lexical material under I and C.

The entire set of zero-level categories in the grammar is then (21), together with I and C. We have 16 categories altogether. We do not assume a lexical feature analysis of zero-level categories. We make no attempt to "collapse" phrase structure rules into fewer ones, in the manner described in section 2.1.2. It does not hurt, though, to assume that the four lexical categories N, A, V, and P have inherent lexical features [±N, ±V] as proposed by Chomsky (1970); we make no proposal regarding the feature analysis that might be given to the other "minor" lexical categories, as well as to the non-lexical I and C.

2.3.2.2 Projections

As in (Chomsky, 1986b), we assume that our X-Bar system is two-level, satisfying uniformity for all categories, except auxiliaries and Cnj. For each zero-level category X, X ≠ Cnj, there are two projections X^1 and X^2, which may also be written X′ and X″, or XB and XP. We adopt the notation XB and XP to denote the first and second projections of X.

It is useful to review the traditional linguistic status of the projections of the no-lexical categories I and C. IP, phrasal projection of inflection, was called S or *sentence* in earlier Extended Standard Theory and Government-binding literature. Base rules expand IP into a subject NP and predicate IB. The category IB corresponds to what is traditionally called a predicate phrase. The category CP, for *clause* or complementizer phrase, was called S′ in earlier work. It contains a sentential form IP, and positions for a complementizer and Wh-movement.

We do not assume that the bar level of a category is one of its syntactic features, as in other frameworks (Gazdar et al., 1985). Instead, according to the X-Bar convention the print name of a category is sufficient to determine the lexical feature analysis of that category, if one is proposed, and its bar level. (An alternative to our assumption, as discussed in section 2.2.1, is to let *Bar* be a feature of all categories. The non-terminal vocabulary of the grammar the reduces of the single two categories Z and X where Z is the start symbol and X is our generic category.)

2.3.2.3 Attribution of categories

We now sketch the association of attributes and syntactic categories in the grammar. Only major attributes are considered. The association defined here is maintained at all levels of representation (see section 2.4).

The principal attributes associated with nouns are *AGR*, *anaphoric*, *pronominal*, and *Wh*. These attributes are associated with and synthesized by N, NB, and NP. *AGR* is the morphological agreement element of nouns, and its value ranges over triples of the form <*Person*, *Gender*, *Number*>, as usual. The attributes *anaphoric*, *pronominal*, and *Wh* are binary-valued. The values of the attributes are lexically specified and set according to the agreement form of the nominal and whether the nominal is pronominal, anaphoric, or a *Wh* form. The attributes *anaphoric* and *pronominal* classify overt NPs as shown in (22). Examples of the three occuring types of NP are are "each other" and "themselves" (anaphor), the personal pronouns (pronominal), and names like "John" (referential). There are no overt instances of [+*anaphoric*, +*pronominal*] NPs.

(22)

	Referential	**Anaphor**	**Pronoun**	**--**
anaphoric	−	+	−	+
pronominal	−	−	+	+

In addition to the four synthesized attributes noted above, NP also bears three attributes *empty*, *RefIndex*, and *Chain*. The attribute *empty* is binary valued and indicates whether the NP bearing it dominates any lexical material; *Chain* is integer-valued and its value is set by the trace convention of section 2.2.3; *RefIndex* is integer-valued and set according to the Binding theory of section 2.6. The attribute *empty* is redundant in the sense that its value is determined by the structure dominated by NP. However, it is convenient to have this attribute on nodes, for use in attribution statements. This permits the language of attribution statements to be limited only to attributes of symbols in the grammar, and to avoid altogether reference to syntactic structure.

An important question arises at this point. If an NP is empty, how are the values of its *anaphoric*, *pronominal*, and *Wh* attributes determined? Even more fundamentally, does it make

sense to attribute features to empty categories and distinguish them by type, as in (4)?[16] There is no lexical material contained by the NP from which the values could be determined. These questions are of theoretical interest in Government-binding. The first one was first addressed by Koster (1978), who assumed, counter to current assumptions, no reason to distinguish trace and PRO. Koster's assumption, though, was part of a larger proposal for the elimination of transformations from the grammar, reducing the trace-antecedent relations to relations of bound anaphora. If we maintain the transformational mechanism for the generation of empty categories, the problem is simplified somewhat, since the values of the attributes in question may be determined at deep structure by the values of the attributes associated with the phrase occupying the position of the empty category.[17] In section 2.6.4 we show a mechanism by which empty (NP) categories may be classified by the structural context in which they appear.

The classification induced by the *anaphoric* and *pronominal* attributes on empty categories is shown in (23). Notice that NP-traces are anaphors and Wh-traces are name-like elements. Wh-traces are regarded in GB grammar as logical variables, filling an argument slot of some predicate, and bound by the Wh-phrase which initially occupied the position of the Wh-trace. The Wh-phrase is interpreted as a logical operator which binds the trace. The elements pro and PRO in (23) are base-generated empty NPs which fill the role of arguments in certain configurations.

(23)

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

The principal attributes associated with verbs are *tense*, *part*, *prog*, *past*, *subcat*, and Θ. The attributes *tense*, *part*, *prog*, *past* indicate tense and aspectual information of verbs, and are associated with and inherited by V, VB, and VP. The attribute *subcat* is the subcategorization frame of the verb, which we had occasion to discuss in section 2.2.2. Although, as discussed in 2.2.2, verbs may specify several kinds on selectional restrictions on their complements, and as we will see in section 2.6 also their grammatical Case, we omit these details here. Thus, for us, a

---

16 This question is worth an analogy. That empty categories may bear features and be distinguished by "types" is no more unusual than that the black holes of the physicists should have mass, age, and different histories.

17 The problem does not disappear, since empty NPs may be base-generated independently and it is possible to have NP-movement followed by Wh-movement. See also section 2.6.4.

subcategorization frame is simply a list of the syntactic categories selected by the verb. A more elaborate scheme may also include specification of some of the syntactic features associated with these categories.

The attribute Θ of verbs is closely related to subcategorization. It may be thought of as a list of place-holders for the argument positions in the logical form of the predicate. For example, if the logical form of the verb *put* is (24.a),[18] the value of Θ is (24.b). Once an interpretation is assigned to a predicate, each argument position of the predicate stands in a certain "thematic relation" to the predicate. We use more mnemonic names, as in (25), to denote the argument positions of a predicate. This issue is discussed in more detail in section 2.7.[19]

(24) a. *put*.LF : PUT(*x*, *y*, *z*)

b. *put*.Θ : <x, *y*, *z*>

(25) a. *put*.LF : PUT(*agent*, *theme*, *location*)

b. *put*.Θ : <AGENT, *theme*, *location*>

Lastly, we consider the syntactic features associated with I and C. The element I carries aspect, tense, and mood indicators associated with sentences. We assume two varieties of binary-valued attributes *tense*, *part*, *prog*, *mood*, *past*, one inherited and the other synthesized. We do not get into this issue here; but see Chapter 3, section 3.5. The category I is empty in typical constructions. In this case, the tense and aspect indicators affix-hop to the verbal element immediately following the I node. The inflectional node also carries the abstract marker *AGR* for morphological agreement. This element must coincide with the same feature of the subject NP. As the tense and aspect elements, *AGR* affix-hops to the following VP, if I is empty.

---

[18] Montague semanticists should read *put'* in place of PUT in (24) and (25).

[19] We ignore the question of redundancy rules in the lexicon. There is a great deal of redundancy in the information contained in the logical form of a predicate, its thematic grid, and its *subcat* frame. It seems likely that the later two may be derived directly form the logical form. See also (Williams, 1981).

The element C bears similar attributes to those associated with I; current Government-binding accounts assume the C node is the target for Subject-Auxiliary inversion (move-I). C also bears binary features *thatcomp*, *forcomp*, *Wh*, and *empty*, which encode the nature of the complementizer that appears under the C node.

2.3.3 The Base of English

In this section we state the phrase-structure rules which we assume throughout this thesis are provided by the base.[20] As with the vocabulary of the grammar, there has been no attempt by GB theorists to provide an explicit set of base rules. Most argumentation in Government-binding concerns the definition of the structural relations discussed in section 2.5 and the manner in which the GB subtheories operate. However, phrase-structure is crucial to understand the relations defined and the operation of the subtheories. There is in fact often controversy regarding the phrase-structure analysis that should be given to particular constructions. We survey below the production set that we assume and briefly mention other analyses which have been proposed, but which we do not pursue.

In the statement of the productions below, we make use of two Extended Backus-Naur Form (EBNF) notational conventions: these are the parenthesis, to denote optionality of a constituent, and the vertical bar '|', to denote alternatives on the RHS of a production. The phrase-structure rules are given without any of the attribution that may be associated with them, since this would be distracting for the purposes of this section. Informally, though, we might make some comments about attributes associated with the symbols in the productions.

2.3.3.1 Sentence and clause structure: The non-lexical categories

The productions for I and C appear in (26). By the first alternative, I and C may dominate no lexical material. In this case, the attribution associated with the productions set the value of the attribute occurence *empty* associated with the category to '+'. The attribution also defines the values of the tense, mood, aspect, and agreement attributes associated with the categories according to context.

---

[20] Providing an explicit listing of the phrase structure rules assumed might seem like a somewhat parochial and tedious enterprise, but yet assumed phrase structure crucially determines the sense of the structural relations defined on derivation trees and the operation of processes that refer to those relations.

(26) a. I → ∈| (M)(HAVE)(*to*)(BE1)(BE2)| DO

b. C → ∈| *for*| *that*| *whether*

The non-empty alternatives in (26.a) introduce the auxiliary system of English, which we assume is generated under the I node, as shown (Chomsky, 1957). The exact nature of the auxiliary is discussed in Chapter 3. The non-empty productions in (26.b) insert the specified grammatical formatives *for*, *that*, and *whether* under the complementizer C. None of these formatives is listed in the lexicon as an item of category C. This is the subtle distinction introduced by Chomsky (1965) between *specified grammatical formatives*, and (regular) grammatical formatives listed in the lexicon.

The productions for the projections of the non-lexical categories I and C appear in (27). This analysis is proposed by (Chomsky, 1986b), and seems to be widely accepted by GB linguists at the present time.

(27) a. IB → I VP

b. IP → NP IB

c. CB → C IP

d. CP → (NP| PP| CP) CB

The rules in (27) follow from the general X-Bar schema for English XP → ZP XB and XB → X YP, where X is head of the phrase XP, YP is complement of X, and ZP is specifier of X. The schema capture the fact that in English complements follow their heads, while specifiers preceed them. The rules (27), with reference to specific categories in the grammar, state the exact range of complements and specifiers for the two head categories we are concerned with, I and C.

Rule (27.a) analyses IB (or predicate phrase) into two parts, the inflectional element I and a verb phrase VP. The element I, carries tense, aspect, and mood features, which are transferred to VP, if I is empty. Where I is not empty, the feature *tense*, if present, is morphologically realized on

the first verbal element dominated by I. Rule (27.b) provides the classical analysis of a sentence IP into a subject NP and a predicate IB. This rule is discussed fully in sections 2.5-2.10, when we get to the GB subtheories. Rule (27.c) associates a complementizer position C with each sentential form IP, and projects to the category CB. This category, in turn, may be projected up to CP (clause) with an optional slot for clausal operators, such as Wh-phrases (27.d). The categories of these operators has been limited to NP, PP, and CP, although we might also allow AdvP.

At this point it is perhaps worthwile to make the discussion less abstract by providing actual examples of the application of the productions (26) and (27). We begin with examples (28.a-c). They illustrate the application of the productions (26.a) and (27.a) to expand the symbols I and IB. Examples (28.d-e) show the use of productions (26.b) and (27.b) to expand C and CB. C may be expanded into the empty category $[_C \in]$ or the specified formative *for*. These two examples show also the use of productions (27.b-c) for IP and CB. Lastly, examples (28.e-f) illustrate the application of production (27.d) for CP, providing an optional empty slot $[_{PP} \in]$ for Wh-movement. This slot would be the target of movement to form the sentence "She wonders where John left for $[_{PP} \in]$."

(28) a. John $[_{IB} [_I \in] [_{VP}$ left for Rome]]

b. John $[_{IB} [_I$ may have] $[_{VP}$ left for Rome]]

c. She allowed John $[_{IB} [_I$ to] $[_{VP}$ leave for Rome]]

d. $[_{CB} [_C \in] [_{IP} [_{NP}$ John] $[_{IB}$ left for Rome]]]

e. She wants $[_{CP} [_{CB} [_C$ for] $[_{IP}$ John to leave for Rome]]]

f. She wonders $[_{CP} [_{PP} \in] [_{CB}$ John left for where]]

The analysis of clause structure just presented is based on (Chomsky, 1986b) and differs from (Chomsky, 1981). In the earlier work, IP and CP are headless categories outside the X-Bar convention. The analysis assumed for them was (29.a-b). Note the categories IB and CB are not present. The structure of the complementizer position was as shown in (12.c), containing two slots, one for Wh-operators, and one for complementizers proper. The reader is referred to

(Chomsky, 1981, 1982) for the motivation of the earlier analysis, and to (Chomsky, 1986b) for the motivation of the analysis we assume here.

(29) a. IP → NP I VP

b. CP → C IP

c. C → (NP| PP| CP) (*for*| *that*| *whether*)

2.3.3.2 Major lexical categories

The phrase-structure analysis that we adopt is adapted from (Jackendoff, 1977). Jackendoff's analysis assumes a three-bar level system, in which complements of a head appear under the single-bar projection, adjuncts under the double-bar projection, and specifers under the triple-bar or maximal projection of the head. Since our analysis uses two-bar levels only, as most current GB literature, we let complements appear under the single-bar projection, as before, and adjuncts and specifiers under the double-bar or maximal projection.[21]

The rules for the projections of N appear in (30). Rule (30.a) is a modified version of Jackendoff's rule for N'. A first point of departure is that we do not allow NP complements of N, but rather assume that they are base-generated as objects of the preposition *of*.[22]

---

[21] We will relax the X-Bar conditions on the places where adjuncts may appear, to include the single-bar projection -- i.e., the level normally reserved for complements. Adjuncts and parenthetical elements are particularly free to intervene between heads and their complements in VPs, as the italicized segments in (i) show. More on this issue in section 2.6. I thank Slava Katz for his help to obtain these data.

(i) a. George William Fairfax, Washington's companion during his frontier surveying days, had brought *home to Belvoir* a tall, slender bride named Sally.
b. It brought *home to me once again* the difficulties of integrating cultures.
c. It is amazing how many men of ability saw *not* the slightest need to volunteer their services.

[22] The question of *of*-insertion is far from resolved in the linguistics literature. Chomsky (1981) discusses four possibilities, depending on whether the preposition *of* is transformationally inserted or base-generated to form the phrase [*of* NP], and depending on the label given to the root of the phrase, NP or PP. A fifth possibility,

(30) a. NB → N (PP| CP| VP)

b. NP → (NP) (ArtP) (QP) (AP) NB (CP)

Due to the conflation of Jackendoff's three-level into Chomsky's two-level system, there are several other points of departure in (30) from the analysis provided by Jackendoff. We will consider only one additional point, namely that rule (30.a) permits VP complements of N. In Jackendoff's analysis this is not permitted since, among other reasons, VP is not a maximal projection. The role of VP in (30.a) is, naturally enough, to allow VP phrases as complements of N, as in (31). Some might argue that the VP-like complement is actually a full infinitival clause CP, with empty inflection, complementizer, and subject. But at this time we do not have any more linguistic motivation for assuming this more complicated analysis than for assuming the VP analysis, and thus choose the alternative with the simpler phrase-structure. Construction (31.a) is superficially similar to the *Poss-ing* construction, which is quite problematic and has motivated radically different analyses of NP (Kornfilt, 1984; Abney, 1987), as discussed below.

(31) a. the child [$_{VP}$ playing soccer]

b. the man [$_{VP}$ hit by the bullet]

Rule (30.b) denotes 32 different productions that introduce NP specifiers and adjuncts. This rule was obtained by collapsing the rules for N″ and N‴ of Jackendoff. The reader is referred to that source for the motivation of the particular analysis. The use of the noun phrase rules (30.a) is illustrated in (32.a-c), and the use of (30.b) in (32.d).

(32) a. [$_{NB}$ report [$_{PP}$ in the drawer]]

b. [$_{NB}$ report [$_{CP}$ (which) we found yesterday]]

c. [$_{NB}$ report [$_{VP}$ found yesterday]]

d. [$_{NP}$ (the) (incredibly many) (tedious) [$_{NB}$ reports on him] (which the CIA gathered)]

---

considered by (Jackendoff, 1977), is that the preposition *of* and the NP do not form a constituent at all, but rather are sisters of the head N. We adopt what seems to be the simplest solution.

Before leaving the subject of NP phrase-structure, we discuss briefly the Poss-ing construction. Consider (33). (33.a) is headed by noun *funding*, which is formed by nominalization of the form *fund*. (33.a) may be generated directly by the NP base rules (30). (33.b), on the other hand, must be analyzed either as a headless construction, or as a construction with an empty nominal head. The formative *funding* is in either case a verb taking an object. We account for (33.b) with rule (34), which is added to the ones in (30). Rule (34) is headless and clearly outside the X-Bar convention. The NP on the right is possesive, while the VP is headed by a progressive verb. The structure generated by (34) is headless; the NP is a specifier, while the VP is complement.

(33) a. $[_{NP}$ [The president's] $[_{NB}$ funding of the Contras]]

b. $[_{NP}$ [The president's] $[_{VP}$ funding the Contras]]

(34) $NP \rightarrow NP\ \ VP$

The Poss-ing construction has created some controversy concerning the structure of NP. Kornfilt (1984) develops an analysis of possessive NPs in Turkish in which the agreement element *AGR* heads NP In Abney (1987), the determiner element D projects up to the maximal projection DP, which is to replace NP throughout the grammar. The category NP appears as complement of D, as in (35), presummably with a much simpler phrase-structure than (30) permits. Both Kornfilt's and Abney's proposals introduce parallelism between the structures of NP and IP; the head element *AGR*/D occupies in NP/DP a position parallel to that of inflection in IP. We do not pursue either of these two analyses of NP. They are mentioned here to show the degree of controversy that may be found in Government-binding regarding phrase-structure. The interested reader is referred to the original sources cited for detailed discussion of the issues involved and consequences of the two proposals.

(35) a. $DP \rightarrow XP\ \ DB$

b. $DB \rightarrow D\ \ NP$

The phrase-structure rules of the verb phrase appear in (36). Rule (36.a) is the analysis of V' which Jackendoff (1977) proposes as a first approximation to verb phrase structure; it is EBNF shorthand for 108 (that is $2 \times 3 \times 3 \times 2 \times 3$) productions. Thes rule is simplified somewhat by

Jackendoff, by use of lexical features to merge some of the complement categories in the manner of (12) above.[23] The rule is has associated with it the subcategorization rule in (14), which defines the value of the *subcat* attribute on the verb. Rule (36.b) is an adaptation of Jackendoff's rule for V'', which corresponds to our VP (Jackendoff's V''' corresponds to our IP). Jackendoff assumes that the elements *have* and *be* of the auxiliary system are generated as specifiers of V. In our analysis, though, they are generated as auxiliaries under I, following Chomsky's original analysis (1957). This is justified by Steele (1982).

(36) a. VB → V (NP) (NP| AP) (AdvP| QP) (PP) (PP| CP)

b. VP → (AdvP) VB (AdvP| PP) (PP) (CP)

Rule (36.a) interacts with the lexicon and the attributed lexical insertion rule (15). Application of the 108 productions implied by the rule is constrained by the *subcat* feature on the head verb. In practice, some of the 108 productions never appear in any derivation; there is no verb which would have the required complement structure (e.g., no verb selects five complements NP-AP-AdvP-PP-CP -- or at least no analysis that I am aware of has ever proposed this). From an empirical point of view, this means that rule (36.a) is not accurate by itself and that its apparent simplicity is gained only at the expense of attribute definitions. The feature *subcat* on most verbs has typically many alternate values. The verb *tell*, for example, has at least five subcategorization values [NP], [NP-PP], [NP-NP], [NP-CP], [PP-CP]. See (37).

(37) a. $[_{VB}$ tell $[_{NP}$ the truth]] / $[_{VB}$ tell $[_{NP}$ Mary]]

b. $[_{VB}$ tell $[_{NP}$ the truth] $[_{PP}$ to Mary]]

c. $[_{VB}$ tell $[_{NP}$ Mary] $[_{NP}$ the truth]]

d. $[_{VB}$ tell $[_{NP}$ Mary] $[_{CP}$ PRO to stop smoking]]

e. $[_{VB}$ tell $[_{PP}$ to Mary] $[_{CP}$ PRO to stop smoking]]

---

[23] Jackendoff's simplified version of the rule for V', his (4.32), is an abbreviation for 2 × 5 × 2 × 5 = 100 productions (ignoring the slot *Prt* for verb particles which he uses).

Rule (36.b) introduces adverbial, prepositional, and clausal adjuncts of V, as in (38). The non-head elements in (36.b) are commonly referred to as adjuncts, rather than specifiers.

(38) a. [$_{VP}$ quickly [$_{VB}$ left]]

b. [$_{VP}$ [$_{VB}$ left] quickly]

c. [$_{VP}$ [$_{VB}$ see him] at the meeting]

d. [$_{VP}$ [$_{VB}$ request new weapons] to promote democracy]

We conclude this section with the phrase-structure rules for projections of P and A. These are shown in (39) and (40). As with our other base rules, these are adapted from Jackendoff's (1977) comprehensive treatment of the X-Bar system.

(39) a. PB → P ((NP) (PP) | CP)

b. PP → (AdvP) PB (CP)

(40) a. AB → A (PP) (PP| CP)

b. AP → (AdvP) AB (CP)

Some examples of the application of the productions appear in (41). Notice that, adopting Jackendoff's proposal, we regard the "adverbs" in (41.a) as prepositions. Also, adjectives do not take NP objects directly (41.c); rather a PP with NP object is assumed, without transformational *of*-insertion.

(25) a. [$_{PB}$ now] / [$_{PB}$ here]

b. [$_{PB}$ to [$_{NP}$ the meeting]] / [$_{PB}$ from [$_{PP}$ under the bridge]]

c. [$_{AB}$ young] / [$_{AB}$ kind [$_{PP}$ of him]]

2.3.3.3 Phrase structure of minor categories

The phrase structure rules of the minor lexical categories Adv, Q, Art, and Deg are defined by the schema (42). None of the minor categories takes complements at the single-bar level. The rule (42.b) for the double-bar projection permits the attachment of adverbial premodifiers.

(42) a. $XB \rightarrow X$

b. $XP \rightarrow (AdvP)\ XB$, for X = Adv, Q, Art, Deg

The syntax of these rules is seldom discussed in the Government-binding literature.

2.3.3.4 Noun compounding and coordination

Noun compounding and coordination are processes which do not bear directly on the central theoretical issues of Government-binding, and hence they have received relatively little attention in the GB literature. However, the two processes must be dealt with in any grammar that hopes to attain even moderate descriptive coverage of the language in question. Traditional treatment of compound and coordinate structures has assumed rule schema with regular expression notation (Chomsky, 1965) to provide recursive nesting, as well as unbounded degree of branching. Such treatment predicts far more ambiguity that is actually found in natural language compounds and coordinates (Langendoen, 1987).

We define the simple "concrete syntax" (43) for the generation of compound and coordinate structures. We distinguish it from the linguistically motivated "abstract syntax" (DeRemer, 1976) that may be computed on the basis of those rules. The computed structure may reflect more closely the semantic relations between the elements in the compound and coordinate structures, as suggested by (Jensen, 1987) and (Langendoen, 1987).

(43) a. $N \rightarrow N\ N$

b. $X \rightarrow X\ (Cnj)\ X$

Rule (43.a) yields binary branching structures outside the X-Bar convention. The number of nouns which may be compounded is limited perhaps only by performance or style factors, and

not by known grammatical principles. The process is illustrated in (44). The place of the rule in the grammar may be debated. It may be argued that the rule is lexical, so it does not belong to the syntactic component. We ignore the issue here. We do not imply that binary branching is the type of branching that should be assigned to all compound constructions in language, but merely that binary branching rules are sufficient to generate all such constructions.

(44) a. [$_N$ test results]

b. [$_N$ research division information assets security report]

Scheme (43.b) defines productions by substituting X for each category in the grammar. As with (43.a), the rules obtained are outside the X-Bar convention. The rules permit coordination of any pair of constituents of the same category, as in (45).

(45) a. [[wine and women] or songs]

b. [wine and [women or songs]]

In spite of the strong limitation that bounded branching imposes on the structures generated, the rules in (43) are highly ambiguous. The number of structural descriptions that they associate with a string containing *n* compounds or conjoins grows exponentially with *n* See (Langendoen, 1987) for the exact figure. When the differences between the meanings of an ambiguous string cannot be attributed to differences in the meanings of the words involved, as in (45), the ambiguity is traditionally attributed to the phrase-structure associated with the string. We assume here that attribute conditions on the application of the rules force one form of branching -- e.g., right-branching. Then, the proper logical form that should be attached to a compound or coordinated string is synthesized by an attribute on the structures defined by (43), in the manner suggested by Jensen (1987).

2.3.3.5 Empty categories: base generation

Certain base rules permit generation of empty categories. The instances of this that we have seen are the productions (26) above, which may generate empty complementizer and inflection nodes. There are many other instances of empty categories which may be base-generated, and for which no productions have yet been introduced. The empty categories in question serve as

landing sites for the application of rule move-α, and may also function as implicit arguments in certain configurations. The rules that introduce them are of the form (46), where XP is a maximal projection like NP, PP, or CP.

(46) $XP \rightarrow \epsilon$

The attribute *empty* associated with I and C is also associated with XP, for each XP to which (43) applies. The corresponding attribute occurrences are positively specified when the rule applies. We now restrict our attention to NP. The attribution (22) and (23) above shows how the attributes *anaphoric* and *pronominal* associated with NP partition the nominals. If we take into account the feature *empty*, we obtain the functional typology (47) of noun phrases (van Riemsdijk and Williams, 1986). The typology (47) is extremely important in Government-binding theory; it is relevant for the operation of the transformational component and most GB subtheories described in the remainder of the chapter.

(47)

	Referential (Wh-trace)	**Anaphor (NP-trace)**	**Pronominal (pro)**	**-- (PRO)**
anaphoric	−	+	−	+
pronominal	−	−	+	+
empty	− (+)	− (+)	− (+)	− (+)

When an NP is overt, the lexical material under it determines its functional type -- i.e., the material determines the values of the features *anaphoric* and *pronominal* synthesized by the NP. When the NP is empty, it is not clear how its functional type may be determined. We might hypothesize that in cases of movement the value of the features on the moved constituent is sufficient to determine the value of the features at the extraction site. However, the theory of transformations alone is not sufficient to distinguish between Wh-trace and NP-trace, assuming that indeed NP- and Wh-movement result from application of the same transformation move-α. The trouble is that a given NP may be applied NP-movement followed by Wh-movement, as in "Who$_i$ e_i was beaten e_i". With the condition suggested, both traces would be wrongly predicted to be of the same type.

It turns out that the various GB subtheories, in particular thematic, Case, and binding, determine the distribution of nominals in phrase structures. It is possible to infer the type of an empty NP from the context in which it appears. This is described in section 2.6 on Case theory. At this stage, it is immaterial whether base rule (33), with X equal N, sets the values of the

anaphoric and *pronominal* features when it applies, or whether the value of the features are left unspecified by application of the rule. In the first case we say that the base generates empty categories with functional type specified. Occurrence of a category at a given structural position, though, is contrained by the GB subtheories. If the second approach is taken, we say that the base generates a "generic" empty category, whose functional type is later determined by interaction with the GB subtheories.

Base-generation of empty categories by rule (46) is needed as part of the mechanism of transformations. To illustrate, the derivation of (48), whose surface structure is (49.a), assumes the deep structure (49.b); both deep and surface structures have an identical number of empty categories.

(48) who do you think that Mary likes?

(49) a. $[_{CP}\ [_{NP}$ who] do $[_{IP}$ you think $[_{CP}\ [_{NP}\ \epsilon]$ that $[_{IP}$ Mary likes $[_{NP}\ \epsilon]\ ]]]]$

b. $[_{CP}\ [_{NP}\ \epsilon]\ [_{IP}$ you do think $[_{CP}\ [_{NP}\ \epsilon]$ that $[_{IP}$ Mary likes $[_{NP}$ who]]]]]

2.3.4 An alternative view of phrase-structure: Binary branching

Brief examination of the phrase-structure rules developed above reveals several descriptive shortcomings. For example, rule (36.b) permits attachment of only a limited number of adjuncts to a given verb, one adverbial and up to two prepositional. But natural language allows a larger number of adjuncts, in any order, as shown in (50). It is not clear apriori that an upper bound may be placed on the number of adjuncts that may be attached to a verb.

(50) The treaty was signed $[_{PP}$ in Paris] $[_{PP}$ on March 22nd] ... $[_{PP}$ by the committee].

Two other descriptive shortcomings of the rules above are that they do not provide an account of free constituent order and occurrence of non-arguments, such as parentheticals, between the complements of a predicate. Even in English, a language characterized by a rigid word order, complements of a verb are relatively free to occur in several orderings, as in (51). A constraint on this freeness is an adjacency condition on Case assignment, which requires that an object be

adjacent to its verb to receive Case. But even this constraint is not strict on a superficial analysis, as shown in (51).[24]

(51) a. Our fathers brought forth [$_{NP}$ a new nation] [$_{PP}$ on this continent]

b. Our fathers brought forth [$_{PP}$ on this continent] [$_{NP}$ a new nation]

Insertion of parenthetical elements and adjuncts between complements of a head is not permited by the base rules developed. This restriction is built into the X-Bar theory, which assigns bar-levels to the kind of constituents that may be attached at a level; complements attach at the single-bar level and adjuncts and specifiers at the second. But with this restriction, base rules are not capable of generating any of the strings (52).

(52) a. She prefers, obviously, the mink coat.

b. It brought home to me once again the difficulties of integrating cultures.

c. I saw hanging above Jack's desk a bachelor's degree.

The treatment of (50)-(52) in Government-binding grammar is varied. Attachement of an unbounded number of constituents is usually dealt with by rule schema using regular expression notation, in particular the Kleene star. Free constituent order and insertion of non-arguments between complements are dealt with by means of syntactic features, transformation, or more often multiple subcategorization frames for the same lexical entry. Jackendoff (1977) uses the feature [$\pm$*trans*] to indicate that a category on the right-hand side of a context-free rule is "transportable," in the sense that its position on the right-hand side of the rule is free or arbitrary.

Non-canonical positioning of NP among the complements of a verb is often described by the non-structure preserving transformation *Focus* NP-*shift*. This transformation may dislocate a noun phrase from verb-adjacent position and Chomsky adjoin it to a position outside the VP

---

[24] The examples are from (Jensen, 1987).

projection, possibly after verbal adjuncts. Occurrence of non-arguments between arguments of a verb is possible by application of NP-shift.

Unfortunately, remarkably little information is found about the NP-shift transformation in the Government-binding literature. The displaced NP may be adjoined to the VP projection of the head verb, or to the IP or CP node of the nearest higher clause. Either of these attachement sites yields a string-equivalent result. Empirical tests, most likely highly theory-dependent, are needed to determine the correct attachement site. More crucially, the conditions under which Focus NP-shift may apply are difficult to specify, risking the possibility of making vacuous the claim of the adjacency condition on Case assignment. The conditions under which NP-shift may apply are not characterized by a sharp transition from satisfaction to violation, but rather by some more graded function. The criteria for satisfaction may be related to "prominence" of the NP in the sentence, although it is also often suggested that the condition is related to "heaviness" of the phrase, e.g., as determined by its morphological length. We return to this issue in section 2.6 on Case theory.

The first two devices noted above, the Kleene star notation and the feature [±*trans*] of Jackendoff, are not available within the context-free formalism assumed for base rules, although they certainly are in extended frameworks. The NP-shift transformation, as we noted, is highly problematic and yields cumbersome analyses for what appears to be relatively simple phrase structure; see section 2.6 below. We adopt as a working hypothesis that non-arguments may be optionally base-generated at the single-bar level, between complements of a verb, and contrary to X-Bar constraints.

Jensen (1987) has proposed dealing with the three descriptive problems noted above by limiting phrase structure rules to a recursive binary-branching form. The empirical effects of the Kleene star, Jackendoff's feature, and the NP-shift rule may be reproduced elegantly and efficiently by recursive binary-branching rules. We illustrate the changes required in phrase structure for complements of V and specifiers of N. Consider replacing rule (36.a) for the single-bar projection of V by the binary-branching (53.a).

(53) a. V → V (NP| AP| AdvP| QP| PP| CP)

b. VB → V

Rule (53.a) is more general and encodes a smaller number (seven) of productions than (36.a). The rule is outside the X-Bar convention, since the head and left-hand side V are at the same bar level. Recursion is possible via the symbol V; each application of the rule may pick up one complement of the verb, to create a higher V node. Due to the possibility of recursion, there is no limit to the number of complements that may be attached and the order in which complements are attached is free. The rule, though, clearly overgeneralizes. Rule (53.a) overgeneralizes in the same way that (36.a) overgeneralizes. Application of both rules must be constrained by the lexicon, via the *subcat* feature of the V node. We attribute (53) as in (54). The attribute condition in (54.a) permits attachement of a constituent XP to the head V if and only if XP is the top element of the subcategorization frame of V. Meanwhile, the attribution removes the top element from the head's subcategorization frame, and propagates this value to the the projected V node. On recursion, the elements of the subcategorization frame of the head V are consumed and production (54.b) applies, satisfying the associated attribute condition.

(54) a. $V \rightarrow V\ XP$

attribution:

$V_0.subcat \leftarrow tail(\ V_1.subcat)$

condition:

$top(V_0.subcat) = XP$

b. $VB \rightarrow V$

condition:

$V.subcat = nil$

The constrained rules (54) solve the descriptive problems of unbounded complement attachment and free constituent order. By a slight change of the attribution in (54.a) it is possible to solve the third problem, namely occurrence of non-arguments at the single bar-level. The change required, in (55), permits attachment of a non-argument without consuming or testing the value of the top element from the subcategorization frame of the head V.

(55) $V \rightarrow V\ XP$

attribution:

$V_0.subcat \leftarrow V_1.subcat$

condition:

adjunct(XP)

Let us now turn to NP specifiers. Rule (30.b), for generation of these specifiers, is an Extended BNF notational abbreviation for 32 ($=2^5$) different productions. At the expense of complicating somewhat the attribution component of the grammar, it is possible to reduce this number of productions to four. The rules in (56) permit recursion on NB in a manner similar to the rules (53). The rules may attach any number of NP specifiers, in any order. Attribute conditions may be added to enforce the correct sequence of attachment of the specifiers.

(43) a. NB → (NP| ArtP| QP| AP) NB

b. NP → NB

A potential drawback of the binary rules (53)-(56) is the higher degree of nesting they assign to the strings they generate. While a number of linguists (Kayne, 1984; Travis, 1984) have argued for binary-branching analyses of the sort produced by (53), the proposals and linguistic motivation for the deeper analyses are controversial and most often not adhered to. Linguistic theory has not found any important constraints on the degree of branching of the structures produced by natural language grammars. Hale (1983) and Kiss (1987) propose, contrary to the binary-branching hypothesis, flat constituent structures (with unbounded branching) as targets for the grammars of non-configurational languages, Walpiri and Hungarian in particular. The proposals eliminate the traditional Subject/Object asymmetry of the configurational languages; they call for attachment of all arguments of a predicate at the single bar-level. Hale introduces a level of lexical structure (LS), related to phrase-structure (PS) by a "linking" rule. The reader is referred to (Hale, 1983) for the details.

The problem noted above of the use of binary rules is not without solution within the framework of augmented phrase structure grammar. As we have argued, this framework is heavily exploited in Government-binding and most current linguistic theories. Jensen (1987) complements her argument for the use of binary rules with a dissociation of the notion of phrase structure, distinguishing two aspects of it. On one hand, *rule structure* is the structure determined by the sequence of application of context-free rules in the derivation of a string. This is the standard language-theoretic sense of phrase structure. On the other hand, augmented phrase structure grammar permits a *computed structure* to be defined and associated, as an attribute, with every phrase in a derivation tree. The computed structure is defined by attribution rules and encodes the possibly flatter structure one might want to assign to the string for purposes of linguistic

description.[25] This technique is illustrated in (57). If (57.a) is not the desired linguistic structure for the string underlying, it is easy to define the computed structure (57.b). Note that the distinction between rule and computed structure that Jensen makes could be viewed as the use of two levels of representation, similar to the use by Hale (1983) of lexical (LS) and phrase structure (PS) in the non-configurational languages.

(57) a. $[_{VB}$ put [the book] [on the table]]

b. $[_{VB}\ [_{V}\ [_{V}$ put [the book]] [on the table]]]

2.3.5 On the elimination of phrase structure rules

Stowell (1981) has called for the elimination of the Categorial (or base) component of the grammar, on the grounds that the productions that comprise it are too idiosynchatic and lead to a base that "suffers from serious problems of explanatory adequacy if it is understood as a hypothesis about the actual structure of linguistic knowledge in the mind." The principal empirical effects of the base, Stowell proposes, can be deduced from a strict adjacency condition on Case assignment.

Under Stowell's proposal, categorial rules contain reference only to the category-neutral variable X^i, in which only the bar-level may be specified. In essence, this limits "phrase-structure" rules to the X-Bar schema (58) (Stowell, 1981).

(58) a. $XP \rightarrow (Spec_X)\ XB$

b. $XB \rightarrow X\ (Comp_X)$

It is clear that the schema (58) contains only a small fragment of the information in the phrase-structure rules of section 2.3.3. For example, the range of possible NP specifiers, VP adjuncts, or P complements is not given in (58). If, as implied by the subscripts in (58), there is a correlation between the symbol X in the schema and the specifiers ($Spec_X$) and complements

---

[25] The notions of "rule structure" and "computed structure" used by (Jensen, 1987) are similar to the notions of "concrete syntax" and "abstract syntax" used in the context of programming languages (DeRemer, 1976).

($Comp_X$) it permits, the simplicity of the schema is only apparent. The relation between X and the allowed set of specifiers and complements is predictable in the case of subcategorization, but not otherwise in general. In the remaining cases, the information may be stated by means of a table. Once this information is added to the schema (58), as an integral part, we obtain only a notational variant, not much shorter, of the set of rewriting rules that could traditionally have been used to represent the sort of information in the schema and the table.

A deeper problem with Stowell's proposal is the fact that the proposed Case-adjacency condition does not predict the order of NP specifiers or any kind of constituents not subject to to the Case filter. This difficulty in the proposal is acknowledged by Stowell. Neither ArtP, QP, or AP is subject to the Case filter condition, so the observed order of these elements in NP specifier position cannot be derived from the adjacency condition. If unspecified semantic factors are to be responsible for the order of constituents, it seems only proper to maintain the traditional context-free rules, or equivalent attributed rules, at least as an observational generalization, until the semantic factors become understood.

The schema (58) may be useful as a basic (X-Bar) format for natural language productions. We illustrate the manner in which (58) may be interpreted as the only two base rules in the grammar, assuming two sets $T_S = \{<X, Spec_X>\}$ and $T_C = \{<X, Comp_X>\}$ of pairs which list the specifiers and complements that each category X permits. This approach seems similar to that of (Dorr, 1987). The non-terminal vocabulary of the grammar thus contains three symbols X, XB, and XP, augmented with lexical features to specify categorial information. The information on the tables is used in (59) to constraint the application of the two productions.

(59) a. $XP \rightarrow (Spec_X)\ XB$

condition:

$< X, Spec_X > \in T_S$

b. $XB \rightarrow X\ (Comp_X)$

condition:

$< X, Comp_X > \in T_C$

(59) illustrates the well-known tradeoff in attributed grammars between using a large vocabulary of non-terminals and few conditions on production application and the converse situation, using a grammar with a reduced non-terminal vocabulary and fewer productions, but each with a large

set of conditions on applications. An extreme case of the latter may be found in Head-Driven Phrase Structure Grammar (Pollard and Sag, 1987), which reduces the rules in a grammar to two, indeed (58). Within the framework of augmented phrase structure grammars assumed, this is indeed possible. The interesting question is the expense incurred, and gain obtained. If the grammar may be stated more concisely and elegantly in terms of attribution functions and conditions on a few rewriting rules than in terms of mostly rewriting rules, then the move is certainly justified.

2.4 Levels of Representation

An important notion in language theory is *membership* of a string in the language generated by a grammar. Language theory usually does not attach significance to the sequence of rewriting steps taken in the derivation of a string, and typically does not impose conditions on the form of such derivations.[26] This is not so in Government-binding grammar. Government-binding identifies certain stages in the derivation of a string and distinguishes them as distinct *levels of representation*, as we discuss in this section.

2.4.1 Derivations

Given a phrase-structure grammar G=(N, T, P, Z), the derivation of a string α in the language L(G) generated by the grammar proceeds by rewriting the initial axiom Z into sentential forms that Z may derive, until α is produced. This is shown in (60). Each step in the derivation, denoted by the symbol $\Rightarrow$, corresponds to application of a rewriting rule in P.

(60) $\quad Z \Rightarrow \ \ldots \ \Rightarrow \alpha$

If the grammar is attributed -- i.e., its symbols are attributed and its rewriting rules are augmented with attribution rules and conditions -- the derivation relation may be extended so

---

[26] A notable exception to this may be found in the L-systems of (Lindenmayer, 1968). An important feature of L-systems, as opposed to phrase structure grammars, is that the rewriting process takes place in parallel; at each step in the process, every symbol in the string is rewritten, according to some production.

that it included reference to application of attribution rules and conditions. That is, we may define the derivation relation as in (61).[27]

(61) Let AG be an attributed grammar with phrase structure component G=(N, T, P, Z), and with packets of attribution rules R and attribute conditions B. Given phrase-markers α and β we say that α *directly derives* β ($\alpha \Rightarrow \beta$) if there exist phrase-markers α_1, α_2, α', and β', such that $\alpha = \alpha_1\alpha'\alpha_2$, and $\beta = \alpha_1\beta'\alpha_2$, and either

(i) the production $\alpha' \rightarrow \beta'$ is in P, or

(ii) there exists an attribution rule $r \in R$ such that $\alpha' \rightarrow_r \beta'$, or

(iii) there exists an attribute condition $b \in B$ such that $\alpha' \rightarrow_b \beta'$

Note that we assume that productions are operating on phrase-markers. In the latter two cases (61.ii-iii), the phrase-markers α and β are identical, except possibly for instantiation or testing of an attribute value in α. We assume that the order of application of attribution rules and conditions is not determined by the order of application of the productions with which the rules or conditions are associated.

2.4.2 Identification of levels

Let $\mathscr{R} = P \cup R \cup B$ be the collection of "rules" and conditions in the grammar. The notion of *level* in the derivation of a string may be introduced by defining an ordered collection of subsets of $\mathscr{R}$, say r_1, r_2, ..., where $r_i \subseteq \mathscr{R}$. We will say that a sentential form σ belongs to level L_i, if there exists a sentential form σ' in level L_{i-1} and $\sigma' \Rightarrow^* \sigma$ by application of rules from r_i alone. We identify level L_0 with the singleton consisting of the axiom Z of the grammar. That is, inductively we have defined a collection of levels L_1, L_2, ..., where L_i is the collection of sentential forms that may be derived from the previous level L_{i-1} by application of rules in r_i .

---

[27] We postpone precise definition of attribute grammars until Chapter 3. We use the symbol $\rightarrow_r$ to denote application of attribution rule or condition *r*.

Notice that the r_i do not have to be disjoint; r_i is the collection of rules allowed in the derivation of L_i. We say that a string σ is *at* level L_i if $\sigma \in L_i$.[28]

As a trivial example of a system with three levels of representation, consider the ordered collection P > R > B, by which we require that all phrase structure rule applications take place before any attribution rule applications, which in turn take place before any attribute condition checking. We may then define three levels of representation PS (Phrase Structure), AS (Annotated Structure), and WS (Well-formed Structure), as in (62).

(62) $Z \underset{(P)}{\Rightarrow \ldots \Rightarrow} (PS) \underset{(R)}{\Rightarrow \ldots \Rightarrow} (AS) \underset{(B)}{\Rightarrow \ldots \Rightarrow} (WS)$

Structures at PS are unattributed phrase-markers, while the structures at AS are annotated with attribute-value pairs. It is possible, though, that some phrase-marker at AS does not meet some attribute condition. Structures at WS, however, meet by definition all attribute conditions. Nothing of consequence seems to follow from the particular definition of levels in example (62). The data dependencies induced by the attribution are probably of more significance; the levels here defined seem to be only convenient notions to talk about the structures generated by the grammar. As we will see in the next two sections, Government-binding grammar defines a more interesting set of levels of representation, and makes important use of them in the definition of the notion of *grammaticality*.

2.4.3 Levels in the Standard Theory

Levels of representation in the Standard Theory are defined by a partition of the set of rewriting rules in the grammar, according to whether they belong to the base or transformational components. In (Chomsky, 1957) and (Chomsky, 1965), *deep structure* is identified as the level that may be obtained by application of base rules alone, including (possibly) lexical insertion. *Surface structure* is the level obtained from deep structure by application of zero or more transformations. The resulting organization of the grammar into levels is shown in (63). The

---

[28] It should be clear that not all ordered collections $r_1, r_2, \ldots$ of subsets of $\mathscr{R}$ will lead to a working (or non-trivial) grammatical system. For example, if r_1 does not contain the production $Z \rightarrow \ldots$ that expands the axiom Z, L_1 and all subsequent levels will be empty. Government-binding grammar uses a slightly more elaborate notion of level of representation, in which levels are not linearly ordered.

examples (8) above show the representations that may be associated with a sentence at these two levels.

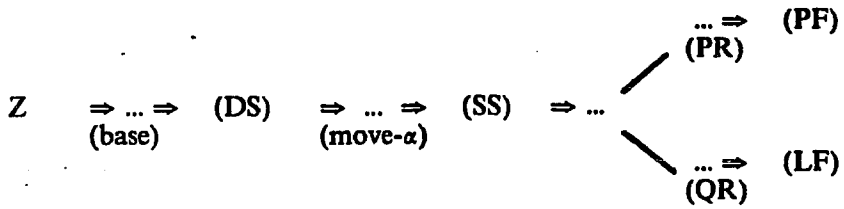
(63) Z ⇒ ... ⇒ (deep structure) ⇒ ... ⇒ (surface structure)
(base) (transformations)

It is assumed in the Standard Theory that the deep structure of a sentence enters the semantic component (which was not specified) for semantic interpretation, while its surface structure enters the phonological component and undergoes phonetic interpretation (Chomsky, 1965). The rules of the phonological component are context-sensitive rewriting rules on attributed/complex symbols. Hence, to the two levels (63) of the Standard Theory we could add a third level *phonetic form*, which follows surface structure and is related to it by phonological rules.

2.4.4 Levels in Government-binding

Government-binding grammar defines four levels of representation, called *D-Structure* (DS), *S-Structure* (SS), *Phonetic Form* (PF), and *Logical Form* (LF), as in (64). The definition of these levels is with respect to a partition of the set of rewriting rules into base and transformational, similar to that used in the Standard Theory. The attribution functions and conditions that define the other components of the grammar, namely by the various subtheories, are assigned independently to apply at one or more of the four levels defined. Every attribution rule and condition of the grammar needs to be supplemented with a statement of the level or levels at which it applies.

(64)



D-Structure is derived by application of base rules alone. We assume that lexical insertion takes place at D-Structure; D-Structures are phrase markers with either lexical items on the leaves or empty categories. S-Structure is derived from D-Structure by application of rule move-α alone. The other two levels, PF and LF in (5), are derived from S-Structure. Phonetic Form results from application of rules in the phonological component, e.g., as in the Standard Theory. We do

not consider PF rules at all here; for our purposes, S-Structure and Phonetic Form are identical.[29] The string generated by a derivation is the yield of the PF phrase-marker produced, and is obtained by debracketization of the PF phrase-marker. Logical Form is an abstract level of representation, removed from the observed surface form of a string. The Logical Form of an utterance is a syntactic encoding (formula) of the meaning associated with the utterance. Logical Form is derived from S-Structure by application of quantifier-raising (QR).

There is no significant difference between the Standard and Government-binding theories in the method they use to define levels of representation. Both refer exclusively to the partition of the phrase-structure component into base and transformational. In GB theory there a slight refinement of the partition in the transformational component, assigning move-α and quantifier raising to different levels. Move-α alone is used in the derivation of S-Structure, while QR is the only rule in the derivation of Logical Form.

We have not related the attribute component of the grammar to the characterization of levels of representation. The GB theory defines levels in terms of the ordering of rewriting rules in (64); attribution rules and conditions are then carefully assigned to apply at different levels of representation (van Riemsdijk and Williams, 1986). The theory specifies, for each attribution rule and condition, the level or levels of representation at which it applies. It permits, furthermore, statement of independent structural constraints on the structures existing at certain levels. Structural constraints are to be distinguished from attribute conditions; the latter are constraints on attribute values.

To illustrate the previous remarks, thematic role assignment takes place at D-Structure, grammatical Case assignment is stipulated to take place at D- and S-Structure, and referential index assignment takes place at LF. The attribution of the trace convention we may assume is applied immediately after application of the rewriting in move-α. Of the attribute conditions to be discussed below, the Case filter applies at S-Structure, after Case assignment, while the Thematic Criterion applies at DS, SS, and LF, in accordance with the Projection Principle. The Empty Category principle is checked at LF, although this is the subject of much debate. Among

---

[29] The assumed identity between PF and S-Structure is an oversimplification. Several style and minor transformations apply at the PF level.

the structural constraints noted above are the Subjacency condition and c-command constraint on movement. We discuss these in detail in the respective sections below.

2.5 Structurally Defined Relations

Several non-standard binary relations between nodes in phrase-structure trees are in common use in Government-binding theory. In this section we define the relations of domination, c-command, government, minimal governing category (MGC), and subjacency. Furthermore, we consider a structural definition of grammatical relations. These relations form part of the vocabulary in terms of which the Government-binding subtheories are stated.

The definition of tree that we adopt is (65). Note that "nodes" may be attributed structures. Except for the domination relation (66) on tree nodes, the other five we consider are non-standard in the terminology of trees and discussed separately in this section.

(65) Let K be a set of nodes and D a set of edges on the nodes in K. A *tree* with node-set K and edge-set D is a directed acyclic graph $T = < K, D >$ satisfying the following conditions:

(i) There exists a node α such that for every other node β there is a unique path $(\alpha, \ldots, \beta)$. We say that α is the *root* of T. (Notice that α is unique.)

(ii) for every node α, a linear order is defined on the set $\{ \gamma : (\alpha, \gamma) \in D \}$ of children of that node. If $(\alpha, \gamma) \in D$, we say that γ is a *child* of α.

(66) Given nodes α and β in a tree T, we say that α *(immediately) dominates* β if there is a path (of length 1) from α to β.

2.5.1 C-command

The c-command relation has a long history, which may be traced back to Langacker's (1969) definition of command (67).

(67) α *commands* β if and only if the first sentence node (IP) that dominates α also dominates β.

Langacker's motivation to introduce the "command" relation was to state the conditions under which a pronominal in a given expression could refer to some other nominal in the same expression. Given the example sentences (68), in which the subscripts *i* attached to the nouns are intended to show that they are coreferential, the problem is to state the conditions under which the grammaticality judgements marked are borne out.[30] Langacker's solution was the pronominalization condition (69), which makes use of the notion "command" he introduced. Given Langacker's assumptions regarding the structure of the sentences in (68), the reader may verify that indeed (67) and (69) yield the judgements listed.

(68) a. If John_i is around, he_i will do it.

b. If he_i is around, John_i will do it.

c. John_i will do it if he_i is around.

d. * he_i will do it if John_i is around.

(69) *Pronominalization Condition*: NP_1 may be used to pronominalize NP_2 unless

(i) NP_2 preceeds NP_1, and

(ii) a. NP_2 commands NP_1, or
b. NP_2 and NP_1 are elements of separate conjoined structures.

Langacker's solution to the pronoun reference problem gives rise to a transformational rule ordering conflict, which motivated several reformulations of the notion "command" in attempts to solve it. We do not discuss this problem here. Instead, we focus on the descendents of the command relation, of which c-command is a particular one. The reader is refered to (Kuno,

---

30 The data and judgements in (68) are from (Kuno, 1987).

1987) for a detailed analysis of the problems associated with Langacker's solution, and the subsequent ones.

Derivatives of the definition (67) of "command" are Lasnik's (1976) notion of "Kommand" (or k-command (Kuno, 1987)), and Reinhart's (1976) notion of c-command (constituent command). These are given in (70) and (71), respectively.

(70) α *kommands* β if and only if the first sentence (IP) or noun phrase (NP) node that dominates α also dominates β.

(71) α *c-commands* β if and only if the first branching node γ_1 dominating α

(i) either dominates β, or

(ii) is immediately dominated by a branching node γ_2 which dominates β and is of the "same" category type as γ_1

The three related variants (67), (70), and (71) of the original "command" relation proposed by Langacker illustrate well the way in which notions and principles of grammar have been developed in the GB framework: by successive refinement. This is not without its problems, since once a number of conflicting definitions for the same concept arise, there is great potential for confusion and imprecision. The problem is aggravated when a particular notion is used in the literature without identifying the intended formulation, or without adhering strictly to the meaning adopted.

None of the three definitions of "command" given above is widely used at this time. The most commonly used definition of c-command in GB theory is (72), which is a revised version of that in (van Riemsdijk and Williams, 1986). This is the definition which we adopt throughout this thesis. It should be noted that (72) does not coincide with either of the definitions of c-command found in (Chomsky, 1981) or (Chomsky, 1986b); we refer the reader to the original sources for the definitions. A related command notion, m-command, which will be used in the definition of government below, is (73).

(72) α *c-commands* β if and only if

(i) α does not dominate β, and

(ii) β does not dominate α, and

(iii) the first branching node γ dominating α also dominates β.

(73) α *m-commands* β if and only if

(i) α does not dominate β, and

(ii) β does not dominate α, and

(iii) the first maximal projection γ dominating α also dominates β.

The "command" relations (72) and (73) are ubiquitous notions in Government-binding. They enter in the definition of government, as well as into the formulation of several constraints on the operation of the movement tranformation, as discussed in section 2.9.

2.5.2 Government

Government is the central structural notion in Government-binding grammar. It is a refined form of the c-command relation (72) and, as in the case of c-command, is a notion that has gone through many different formulations. There are three related notions of government in use in GB theory; a core notion, exceptional government, and proper government. Each of these is discussed in turn.

2.5.2.1 Core notion

The definition of government involves choice of the set of *governors* (nodes in the domain of the relation), of the set of *governees* (nodes in the range), and of the structural condition of the relation. As a first approximation, (74) is the definition proposed in (Chomsky, 1981).

(74) α *governs* β if and only if

(i) α is a zero-level category, and

(ii) α c-commands β, and

(iii) for each node γ c-commanding β, either γ c-commands α or β c-commands γ.

The government relation (74) obtains between a head and each of its complements, as in (75). This relation is required for the processes of Case-marking and thematic role assignment. In (75.a), the head verb *put* governs each of the complements NP and PP, and Case-marks and assigns a thematic role to them. We return to these processes in sections 2.6. and 2.7 below. Similarly, in (75.b) the preposition *in* governs and Case-marks its sole complement NP.

(75) a. $[_{VB}$ put $[_{NP}$ the book] $[_{PP}$ in the box]]

b. put [the book] $[_{PB}$ in $[_{NP}$ the box]]

The definition of government is extremely problematic; the relation is highly dependent on assumed phrase-structure analyses and enters into the formulation of a large number of principles and feature assignment processes in the grammar, each with its own requirements. To illustrate the kind of problems that arise, notice that given definition (74) and the phrase-structure rules of section 2.3 on X-Bar theory, the inflectional element I does not govern its Subject (or specifier) position. This is contrary to the government relation intended in (Chomsky, 1981). The first branching node dominating I is IB, which does not dominate the Subject. Hence I does not c-command or govern Subject position. However, assuming the older analysis (29) of IP assumed in (Chomsky, 1981), the government relation (74) between I and its Subject obtains.

Condition (74.i) needs refinement, as the only condition it imposes on a node to serve as governor is its bar level. But this is not accurate. The intention of (74.i) is that only the categories N, V, A, and P may be governors; that is, those marked by the lexical features [$\pm$N, $\pm$V] . The category I may also be a governor, but only if it contains an *AGR* marker (i.e., if I.*AGR* $\neq$ *nil*). Given that the notion of government is defined as a relation on tree *nodes*, it is

not appropriate to include *AGR* in the set of potential governors, as done by (Chomsky, 1981). *AGR* is a *feature* on the I node, and hence not a potential governor -- unless the domain of the relation is extended to attributes, or unless *AGR* is identified with and treated as a lexical category. Chomsky identifies *AGR* with PRO (= NP[+*empty*, +*anaphoric*, +*pronominal*]) but the treatment of *AGR* in the theory is otherwise as a feature on syntactic categories.[31] This is clearly a minor detail or technicality, but which can easily lead to some confusion. Below we augment condition (74.i) with reference to the syntactic features of a potential governor.

Chomsky (1986b) pursues a different approach to government, relying on a new notion that certain nodes in phrase-structure are *barriers* to government. We do not pursue the development of the new notion here, or the proposed approach to government. The definition (76) of government that we adopt is adapted from (van Riemsdijk and Williams, 1986) and is close to that in (Chomsky, 1986b).

(76) α *governs* β if and only if

(i) α is a zero-level category (if α = I, then *AGR* is present), and

(ii) α m-commands β, and

(iii) each maximal projection γ dominating β also dominates α.

The definition of m-command is (73) above. Under definition (76), a head governs each of its complements and specifiers. The instances (75) of government are accounted for, as are also the cases (77), not previously accounted for. In (77.a) I governs the Subject NP, while in (77.b) it does not. As mentioned above, the *AGR* feature must be present in I; otherwise the specifier position is not governed.

(77) a. $[_{IP}$ Mary $[_{IB}$ $[_{I}$ +*AGR*] $[_{VP}$ left the room]]]

b. * $[_{IP}$ Mary $[_{IB}$ $[_{I}$ *to*] $[_{VP}$ leave the room]]]

---

[31] Government-binding thus blurs the distinction between syntactic categories and syntactic features in its treatment of *AGR*, in the manner of unification-based approaches to grammar (Shieber, 1986).

2.5.2.2 Exceptional government

One instance of government not covered by (76) is illustrated in (78). In (78), the Government-binding analysis is that the matrix verb governs the clausal complement, and also the embedded Subject. It is furthermore assumed that the verb Case-marks the Subject. This is shown clearly in (78.b), where the grammatical Case of the subject is accusative, rather than nominative as in a more typical Subject position. Since the instances of government in (78) do not follow from the core definition (76), something else must be said. In (78), both CP and IP are maximal projections, and thus barriers to government under clause (iii) in (76).

(78) a. Mary wants $[_{CP}\ [_{IP}$ John $[_{IB}$ to leave the room]]]

b. Mary wants $[_{CP}\ [_{IP}$ him $[_{IB}$ to leave the room]]]

Exceptional government in (78) depends on lexical properties of the matrix verb, and certain syntactic properties of the complement clause. Not all verbs may govern and Case-mark the subject of their clausal complement, as shown in (79). The lexical property that permits a verb to exceptionally govern a subject is highly idiosynchratic; we assume here that it is simply marked in the lexicon by means of the feature [±*ecm*] (exceptional Case-marking). This feature is positively specified for those verbs which may exceptionally govern (and possibly Case-mark) a subject. Examples of these verbs include *appear, believe, seem, want*, and several others.

(79) a. * I think $[_{CP}$ the man to be a good fellow]

b. * I concluded $[_{CP}$ the proof to be correct]

The reasons behind the ungrammaticality of the examples in (79), under Government-binding assumptions, are explained in detail in the section 2.6. For the time being, it suffices to observe that neither of the matrix verbs in (79) governs the embedded Subjects, due to lack of the feature *ecm*. The syntactic properties of the complement clause that permit exceptional government are lack of *tense* (in the I node), and absence of an overt complementizer (under the C node). If the embedded clause is tensed, then the embedded I node governs the subject position in the core sense (76), as may be observed in (80.a), where it is clear that the Case assigned to the embedded

subject is nominative, rather than accusative. If an overt complementizer is present, exceptional government is also blocked (80.b-c).[32]

(80) a. I believe [$_{CP}$ he is a good fellow.]

b. * I want [$_{CP}$ that [$_{IP}$ he to be an old fellow.]]

c. I want [$_{CP}$ for [$_{IP}$ him to win the race.]]

A second instance of exceptional government is found in (80.c), where the complementizer *for* governs Subject position. In that configuration, the presence of the complementizer blocks exceptional government by *want*, as in (80.b). Chomsky (1981) assumes that *for* may govern and assign Case to the subject, presummably in an exceptional manner, as done by a [+*ecm*] verb. A condition, as before, is that the complement IP is untensed.

The relation of exceptional government may be defined as in (81).

(81) α *exceptionally governs* β in either of the configurations (i) or (ii),

(i) ... α [$_\gamma$ [[$_\delta$ β ...

(ii) ... α [$_\delta$ β ... ,

where γ = CP, δ = IP, and α is [+*ecm*] in (i) or is the complementizer *for* in (ii).

The formulation of exceptional government in (17) seems fairly ad-hoc, but at the same time it appears to capture exactly what is proposed. We know of no more principled formulation, and leave the issue as just given.

An formulation of exceptional government, attempting to reduce it to the core definition (76) of government, admits of several variants. A possibility proposed in (Chomsky, 1981) is to assume

---

[32] The explanation of the facts in (80) is not so straightforward, since there are independent relations between I and C that must be stipulated; the complementizer chosen determines whether the embedded clause should be tensed.

that a node CP may be deleted if it follows a [+*ecm*] verb. This removes a maximal projection, barrier to government. It is clear that this deletion transformation is not a structure-preserving transformation. In (Chomsky, 1981) CP (his S') is taken a projection of IP (his S), and IP does not count as a maximal projection. But under the phrase-structure analysis of CP in (Chomsky, 1986b), adopted here, IP is the maximal projection of I, and hence also a barrier to government in the sense of (76.iii). Saving Chomsky's proposal would require the more problematic deletion of the IP node, and probably also of CB.[33]

Van Riemsdijk and Williams (1986), adopt a different position, proposing that [+*ecm*] verbs may "mark" the root of their CP complements "transparent" to government (in the sense that they do not count as maximal projections). Presumably this option can be carried to IP, to eliminate its barrierhood. We will ignore these more principled proposals, keeping the ad-hoc formulation (81) of exceptional government, which is an instance of government.[34]

2.5.2.3 Proper government

A last notion of government, which was introduced recently into the theory and figures prominently in GB grammar, is proper government. Its definition is given in (82), using the formulation of (Chomsky, 1981).

(82) α *properly governs* β if and only if

(i) $\alpha = X^0$ or is trace-coindexed with β, and

(ii) α c-commands β, and

(iii) Each maximal projection ϕ which dominates β also dominates α, and

(iv) $\alpha \neq AGR$ (we read $\alpha \neq$ I)

---

[33] That IP is a problem is apparent in (Chomsky, 1986b), where explicit reference to IP is needed, stipulating that it does not count as a barrier (for government) (See his (26)).

[34] Exceptional Case-marking and government are, in any case, peripheral and marked options in a few languages, like English.

Notice that proper government introduces a new class of categories as potential (proper) governors. This class consists of all categories which can receive a trace index, due to movement. This admits moved NP, PP, and CP among others. Proper government also eliminates I as a potential proper governor, although I still counts as a governor in the core sense (76). Condition (82.iv) accomplishes this; it is actually a parameter of universal grammar: (82.iv) is present in the definition of proper government in the grammar of languages like English, and absent in the grammar of so-called pro-drop languages, like Spanish and Italian.

Proper government is the structural relation that enters into the formulation of the Empty Category Principle (ECP), a constraint on the distribution of empty categories. The formulation of the ECP is given in section 2.10. As with the other structural notions of grammar we have discussed, proper government is problematic, and has received many different formulations in different works (van Riemsdijk and Williams, 1986; Chomsky, 1986b).

2.5.3 Minimal governing category

A structural notion derivative from the notion of government, and crucial in the formulation of the binding theory, is that of minimal governing category (MGC). This is a binary relation between nodes in trees, as defined in (83).

(83) α is a *minimal governing category* of β (MGC(β)) if and only if

(i) α dominates β, and

(ii) there exists a governor γ of β, such that α dominates γ, and

(iii) α dominates a SUBJECT accessible to β.

(iv) α is minimal -- i.e., for every other node α' satisfying (i)-(iii), α' dominates α

Conditions (83.i-iii) define the notion *governing category* of (Chomsky, 1981). Condition (83.iv) imposes the minimality condition on the node selected. The notion of SUBJECT in (83.iii) refers to the elements (i) Subject (NP) of an infinitive clause or a noun phrase, (ii) *AGR* (I) in a tensed sentence, and (iii) the NP in a "small clause." Huang (1982) claims that the

SUBJECT-accessibility requirement (83.iii) applies to a category β if β is an anaphor, but not if it is a pronoun.[35] The relevance of notion (83) to the binding theory is discussed in section 2.8.

2.5.4 Subjacency

The next of the structurally defined relations that we consider is subjacency. Subjacency is the *locality condition* (84), which is parameterized by the choice (85) of bounding nodes.

(84) α is *subjacent* to β if and only if the path (X, ..., β), excluding the end nodes X and β, contains at most one bounding node; X is the highest node of the path (α, ..., β).

(85) IP and NP are *bounding nodes* (for English).

Given the choice (85) of bounding nodes, we obtain for the partial phrase markers in (86) that the subjacency relation holds between the nodes marked *i* and *j* in the (a), (b), and (d) cases, but not in (c) and (e).

(86) a. [Orders]$_i$ were given [ϵ]$_j$.

b. [ϵ]$_i$ seems [$_{IP}$ [John]$_j$ be forgotten.]

c. [ϵ]$_i$ seems [$_{IP}$ Bill believes [$_{IP}$ [John]$_j$ to be forgotten.]]

d. [Who]$_i$ did [$_{IP}$ John see [ϵ]$_j$]

e. [Where]$_i$ did [$_{IP}$ John see [$_{NP}$ the man form [ϵ]$_j$]]

Subjacency is the notion involved in the statement of an important condition on the operation of move-α, the Subjacency condition, discussed in section 2.9. The choice (85) of bounding nodes affects the distance over which movement of a phrase is possible in one application of move-α. For Italian and Spanish, the bounding nodes defined by the grammar are CP and NP.

---

35 I thank Prof. Kuno for this observation.

2.5.5 Grammatical relations

The traditional grammatical relations *Subject* (of a sentence) and *Object* (of a verb) are defined structurally in the Government-binding framework, as in (87). The notation [X, Y], where X and Y are grammatical categories, is used to mean "the node of category X which is immediately dominated by Y"; where there is more than one occurence of X under Y, we distinguish them by enumeration, as in [X_i, Y] (Chomsky, 1965).[36]

(87) a. Subject-of-I:
[NP, IP]

b. Object-of-V:
[NP, VB]

The Subject of a sentence is the NP immediately under the IP node of the sentence. Similarly, the (first) Object of a verb is the (first) NP complement of the verb. Definition (87) assumes the phrase structure rules of section 2.3.4. The relations Subject-of-I and Object-of-V may be generalized to an arbitrary category X, by parameterizing (87) with respect to this category. This permits the interpretation of the NP specifier *John* in (88) as Subject of the noun *refusal*, and of the phrase *the offer* as Object of the preposition *of* (or of the same noun *refusal*, assuming transformational *of*-insertion).

(88) $[_{NP}$ John's $[_{NB}$ refusal of [the offer]]]

We note that the Subject relation may be defined in terms of Case or thematic role: Subject is the position at which nominative Case and thematic role may be assigned.

---

[36] The terms "grammatical relation" and "grammatical function" are used interchangeaby. It should be noted that the relation Object-of-V is traditionally written [NP, VP], obscuring the fact that VP does not immediately dominate the Object (under X-Bar assumptions).

2.6 Case Theory

Case theory is the *subtheory* of Government-binding grammar concerned with the distribution of grammatical case in syntactic representations. Traditional grammar identifies case as property of nominals. The inflection of a nominal is determined, in part, by the grammatical case that it bears. In English we observe three inflected forms of nominals, in the personal pronouns. The three inflections observed in (89) for each pronoun are, respectively, the *nominative, accusative,* and *genitive* inflections. English is inflectionally weak; it does not show case-inflections on most other nominal forms and does not distinguish morphologically the cases *accusative, objective*, and *oblique*. These are marked in languages with richer inflectional morphology.

(89) a. I, me, my/mine

b. he, him, his

c. we, us, our/ours

Given these general remarks, we assume a multi-valued abstract syntactic feature *Case*, associated with nominals, and which influences the morphological shape of the nominal. The function of the feature in the theory has three different aspects: Case assignment, Case realization, and Case conditions. We discuss each of these in turn.

2.6.1 Case Assignment

The Case that is appropriate for a nominal in a sentence depends on the position of the nominal. Nominative Case is typically associated with subject position (90.a). Accusative or objective Case are associated with object position of a verb or preposition (90.b), and genitive with the specifier or determiner position in a noun phrase (90.c).

(90) a. *he* wrote the report.

b. Mary wrote (to) *him*

c. *his* father wrote him a letter.

2.6.1.1 Structural Case

Case is assigned to a nominal (within the GB framework) in one of two fashions: Structurally or inherently. The instances of structural Case assignment are (91.a-d). The only instance of inherent Case assignment discussed here is (91.e) (Chomsky, 1981).

(91) a. NP is *nominative* if governed by a tensed I.

b. NP is *objective* if governed by V with subcategorization frame [NP ...]

c. NP is *oblique* if governed by P.

d. NP is *genitive* if it appears in NP specifier position ($[_{NP}$ _ ...]).

e. NP is inherently Case-marked by V in double-object constructions.

The four variants (91.a-d) of structural Case assignment take place at S-Structure under government. In (91.a), the inflection category I assigns nominative Case to its Subject, if it is tensed. Where I is not tensed, the Subject may still be Case-marked, exceptionally, as detailed below. (91.b-c) are the most typical instances of Case assignment. The theory identifies some grammatical categories as *potential Case assigners*; each of them may assign, under government, a particular Case to its NP complement. We assume the feature *Case*, associated with nominals as an inherited attribute, is also associated with potential Case assigners, but now as an inherent attribute. The *Case* of a potential Case assigner is listed in the lexicon. We say that a potential Case assigner X *Case-marks* a constituent Y if the value of the X.*Case* is assigned as the value of Y.*Case*; we write Y.*Case* ← X.*Case*. Instance (91.d) of Case assignment occurs under government by the head noun, although the noun is not assumed to be the Case-assigner.

Chomksy (1981) assumes that the potential Case assigners are the zero-level [−N] categories in the X-Bar system, hence V and P.[37] In addition to these, the inflectional category I is also a Case assigner, provided it is tensed. We assume that the domain of the feature *Case* is (92). The last

---

[37] (Chomsky, 1986a) adds the categories N and A to the class of potential Case assigners. All major lexical categories are Case assigners in this situation.

value, *nil*, is used to represent the situation when, in common terminology, a Case assigner is said to assign no Case (or have no *Case* feature).

(92) *Nom, Acc, Dat, Oblq, Gen, ..., nil*

The *Case* value associated with a Case assigner of category X is often directly correlated with X. We might hope that the *Case* value may be defined for each lexical class X. However, the correlation between the category X and its *Case* is not complete. It must be assumed that the value of the feature is specified separately for each entry in the lexicon. In English, it is necessary to list minimally whether a verb is a Case assigner or not. Verbs like *believe*, *like*, and *want* may assign accusative Case to (the Subject of) their clausal complement, while the verbs *appear* and *seem* do not. Neither do the non-causative forms of verbs like *break* and *open* assign Case to their (deep) Object.[38] Verbs typically assign accusative Case and prepositions oblique. However, verbs may assign other Cases. This is illustrated by the contrast in (93), from German. The grammatical Case that a verb assigns is often related to the "thematic role" that the complement plays in the event or state described by the verb.

(93) a. Ich liebe ihn (accusative)

b. Ich verzeihe dem Mann (dative)

The association of *Case* values with potential Case assigners also depends on the morphology of the Case assigner in question. Passive verbs, which are formed by lexical rule, do not assign Case. Notice that it is crucial to define the level of representation at which passive forms are derived. GB theory assumes that the passive morpheme *-en* is transformationally affixed to a verb in the PF component, by an affix-hopping transformation which dates to the analysis of the auxiliary in (Chomsky, 1957). This is discused in Chapter 3, section 3.5. However, Case and

---

[38] It is still an open issue whether sentences like (i.a) are base-generated or produced by Object to Subject raising, as in (i.b) (cf. Williams, 1981; Levin, 1985). Apparently the choice between the two analyses depends on lexical rather than syntactic factors. See also section 2.7.

(i) a. The window broke.
b. [The window]$_i$ broke [$_{NP}$ e]$_i$.

(external thematic role) must be suppresed in the verb at D-Structure, for otherwise the verb has the wrong set of features at D- and S-Structure. Given that the relation between the Case assigners and the Cases they assign is not regular, we ignore at this time whether the Case assignment properties of an item may be predicted by general lexical rule, like the passive formation rule, or whether these properties have to be listed independently in the lexicon. Most likely, a theory of markedness that combines both methods is required.

The last form of structural Case assignment in (91) is the genitive rule (91.d). This rule simply stipulates that genitive Case is assigned in the configuration $[_{NP}$ _ ...]. No node is involved in the process as a Case assigner. Chomsky (1986a) discusses the possibility of letting N be a potential Case assigner, like V, P and tensed I, so that the NP in specifier position is Case-marked by the head noun.

2.6.1.2 Inherent Case

Inherent Case assignment (91.e) differs from structural Case assignment in that it takes place at D-Structure. The only possiblity discussed in (Chomsky, 1981) is Case-marking of a verb's second object, as in the dative-shift construction (94.b).

(94) a. John gave [a book] [to Mary]

b. John gave [Mary] [a book]

The exact mechanism responsible for inherent Case marking is a highly controversial issue, on which no concesus seems to exist. (Chomsky, 1981) says that "inherent Case is presumably closely linked to θ-role ." Under this presumption, the subcategorization frame of a verb, in addition to specifying the syntactic category of its complement, specifies the particular Case that the complement bears. Inherent Case marking is the basic mechanism for Case assignment in the non-configurational languages, where all complements are generated with with various Case-marking particles attached to them (Stowell, 1981).

A second alternative found in (Chomsky, 1981) is that the first Object and the verb in (94.b) form a constituent, say VB, which Case-marks the second Object, as the analysis (95) shows. The category VB is hence also admitted as a governor and potential Case assigner. The same proposal is advanced by (Stowell, 1981), but now hypothesizing that the constituent [give Mary]

is of category V and is formed at a level below the syntax, by a word formation rule. The word "give Mary", of category V, is then the Case-marker of the second Object.

(95) John $[_{VB}\ [_{VB}$ gave [Mary]] [a book]]

The analysis (95) saves the adjacency condition on Case assignment (Stowell, 1981). This condition is discussed in section 2.6.3 below. A problem of explanation of the proposed analysis is that it does not explain why the constituent analysis (95) is available only for V-NP sequences, and not for sequences where the second element is of some other category, like AdvP or PP. If explicit reference to NP is not included in the solution, then it could be invoked to save the ungrammatical (96).

(96) a. * Carlos $[[_{V}$ gave [to John]] the ball]

b. * John $[[_{V}$ thinks strongly] many ideas]

We have discussed some problems associated with dative-shift constructions. We adopt the first inherent Case marking approach in (Chomsky, 1981). For an older transformational account of the dative-shift construction the reader is referred to (Jackendoff, 1977). We describe our formalization of the Case marking process in Chapter 3. In section 3.3 we give the explicit set of attribution rules that define (by copying) the value of the feature *Case* associated with nominals and various other categories.

2.6.1.3 Exceptional Case assignment

Structural and inherent Case marking are the core instances of Case assignment. In English, there is a marked option for exceptional Case marking (ecm), parallel to that of exceptional government. This option permits certain lexically marked verbs to Case-mark the Subject of their clausal complement. The option permits, also, the complementizer *for* to assign oblique Case to the subject following. The two instances of exceptional Case marking are formulated in (97). Notice that the statement is completely parallel to the definition (81) of exceptional government.

(97) α *exceptionally Case marks* β in either of the configurations (i) or (ii),

(i) $\ldots\ \alpha\ [_{\gamma}\ [\ [_{\delta}\ \beta\ \ldots$,

(ii) $\ldots\ \alpha\ [_{\delta}\ \beta\ \ldots$,

where $\gamma = \text{CP}$, $\delta = \text{IP}$, and α is [+*ecm*] in (i) or is the complementizer *for* in (ii).

The nodes CP and IP in (97) are subject to the same conditions as in (81), namely, the IP node is untensed and the CP clause is headed by an empty C. The phrase "α (exceptionally) Case marks β" in (97) is to be interpreted as "the value of α.*Case* is assigned to the attribute occurrence β.*Case*." The formulation of exceptional Case marking is subject to the same problems and details of realization that exceptional government has. We do not repeat these here.

Since in (97) (implicit) reference is made to the synthesized *Case* feature of the *ecm* verb, it is possible for a verb to exceptionally govern a Subject, while it does not assign (non-*nil*) Case to it. This obtains when the the verb is [+*ecm*], but its *Case* feature has the value *nil*. This is illustrated in (98) with the verbs *seem* and *appear* (98.a), and with the passive *ecm* verb in (98.b).

(98) a. John_i seems/appears $[_{\text{CP}}\ [_{\text{NP}}\ \in]_i$ to be in love]

b. John_i is believed $[_{\text{CP}}\ [_{\text{NP}}\ \in]_i$ to be in love]

2.6.2 Case Realization

The grammatical Case assigned to a noun phrase is the value of the abstract syntactic feature *Case*. The manner in which this feature is morphologically or phonologically realized by the phrase is an independent matter, highly language dependent. *Case* is realized only in nominals. In English, the feature is not overtly realized in most nominals, but only in some of the pronominal forms. Also, the *Case*-values accusative, objective, and oblique are realized in the same way.

The manner in which Case realization is seen depends on assumptions about the lexical items inserted under pre-terminals. If lexical items are inserted in their root (or lemma) form, then *Case* realization determines the spelling or sound given to the item, depending on its *Case*-value. For example, *he*[+*Acc*] would be spelt *him*, etc. On the other hand, if lexical items are inserted in fully inflected form, we must assume that they carry an inherent *Case* feature, corresponding to the form of the inflection. Case "realization" would be in this situation a matter of an attribute condition which requires match between the *Case* features of the inflected form and the node under which it is inserted.

2.6.3 Case Conditions

Case theory constrains the distribution of overt nominals in sentences, according to the Case filter below. Furthermore, Case theory imposes an adjacency condition on Case assignment.

2.6.3.1 Case filter

The condition that Case theory imposes on the distribution of nominals is the so-called *Case filter*, whose statement is given in (99) (Chomsky, 1981).

(99) **Case Filter:**

* $[_N \alpha]$, if α includes a phonetic matrix and N has no *Case*

Since GB grammar postulates four different levels of representation, it is necessary to state the level (or levels) at which the Case filter applies. The Case filter (99) applies at S-Structure, after structural Case assignment takes place at S-Structure.[39] A non-terminal sequence α is said to include a phonetic matrix if its yield is not the empty string. The Case filter marks ill-formed all those strings containing non-empty noun phrases, but which lack grammatical Case (*Case*≠*nil*).

The grammaticality judgements (100) follow from the Case filter. (100.a) has a *Case*-less Subject since the sentence is untensed. In (100.b) the Subject of the embedded clause is Case-marked accusative by the *ecm* matrix verb, satisfying the Case filter. In (100.c), though, the matrix verb *think* is not [+*ecm*] so it does not Case-mark the embedded Subject. Finally, in (100.d) the

---

[39] The notion of *derivation* in Government-binding grammar assigns evaluation of attribution rules and conditions to certain levels of representation, as discussed in section 2.4.

passive verb *broken* has its Case suppressed so the object remains *Case*-less, in violation of the Case filter.

(100) a. * [$_{NP}$ He] be here.

b. I want [[$_{NP}$ him] to be here].

c. * I think [[$_{NP}$ he/him] to be here].

d. * it was broken [$_{NP}$ the vase] by John.

The Case filter is crucially involved in the explanation of the grammaticality of a large number of constructions. It is one of the grammatical principles that lends a high degree of modularity and depth of explanation to the Government-binding theory. We illustrate this point with an account of the English passive construction. The elements of thematic theory needed for the explanation are covered in section 2.7 below. In early transformational grammar (Chomsky, 1957, 1965), passives were derived from active underlying forms by means of the explicit *passive transformation* (101). This account was favored at that time since the similarity of meaning between the active and passive could be accounted for by the identity of their deep structures.[40]

(101) **Passive Rule:**

$$X\ NP_1\ Aux\ V\ NP_2\ Y \rightarrow X\ NP_2\ Aux\ \textit{be}\ \textit{en}\ V\ \textit{by}\ NP_1\ Y$$

The passive rule (101) (after Chomsky, 1957) shows in painful detail the manner in which a passive sentence is transformationally derived from its "deep" active counterpart. The deep form (102.a) is generated by application of (context-free) base rules alone. Form (102.b) is produced by the passive rule (101); after the necessary tense and affix-hopping transformations apply, the surface form (102.c) is obtained (see Chomsky, 1957).

(102) a. John break [$_{NP}$ the vase]

---

[40] In Standard Theory, deep structure enters the semantic component for semantic interpretation and is closer to the representation of the meaning of utterances. An active sentence and its passive counterpart share identical meanings because of a common deep structure.

b. the vase *be en* break by John.

c. the vase was broken by John.

The Government-binding account of the passive differs sharply from the previous one. The passive construction is the result of an interaction between lexical properties of passive verbs, the rule move-α, and the Case and thematic theories. For the derivation of sentence (102.c), we begin with the lexical entry (103.a) for the transitive verb *break*. The verb has *subcat* value [NP] and assigns accusative Case to its object. The thematic grid of the verb (see section 2.7) selects two arguments for the verb, which play the roles *agent* and *theme*. The underlining of the *agent* role indicates that the role is assigned externally, in Subject position. Entry (103.a) is what the lexicon provides. The *passive formation* rule in the lexicon may apply to (103.a) and derive the passive form (103.b). A second feature of the passive rule, in addition to changing the morphology of the verb by attachment of the passive morpheme *-en*, is suppression of the Case assignment ability of the verb, so that *Case* is *nil* in (103.b). Passive formation also suppresses the external thematic role of the verb.

(103) a. **break**
subcat : [NP]
Case : *Acc*
Θ : (*agent*, *theme*)

b. **broken**
subcat : [NP]
Case : *nil*
Θ : (*theme*)

The derivation of (102.c) begins with D-Structure (104.a). The passive form *broken* of the verb is base-generated, as is the optional agentive *by*-phrase. Notice that the Subject position is not overt at D-Structure, and that the surface Subject *the vase* appears in object position at D-Structure. As explained in section 2.7, the thematic role *theme* of the passive verb is assigned to the NP in object position, with all conditions on thematic role distribution satisfied. Thus

(104.a) is a valid D-Structure.[41] The structure is also a potential S-Structure, derived by application of no transformation. However, as an S-Structure (104.a) violates the Case filter (99). The verb assigns no *Case* to its object, since the *Case* is suppressed (lexically) by the passive rule in the lexicon. Hence, in (104.a) the object bears no Case after structural Case assignment and the Case filter is violated. (104.a), though, may be used to derive a well-formed S-Structure. Application of move-α to the deep object yields S-Structure (104.b). Since no thematic role is assigned to Subject position, the moved NP still bears one θ-role. Also, the NP is now Case-marked nominative by the *tense* element in I, satisfying the Case filter.

(104) a. $[_{NP}\ e]$ was broken $[_{NP}$ the vase] (by John).

b. $[_{NP}$ the vase$]_i$ was broken $[_{NP}\ e]_i$ (by John).

This concludes our account of the passive. Neither move-α nor the Case or thematic theories are elements of grammar designed only to explain the passive. The account is highly modular.

2.6.3.2 Adjacency condition

Case theory stipulates the adjacency condition (105.ii) on the configuration for Case assignment (Stowell, 1981). This condition is assumed in GB grammar (Chomsky, 1981; Stowell, 1981), but it is problematic, as it is subject to some systematic and several idiosynchratic exceptions.

(105) **Case Adjacency Condition:**
In configuration [... α ... β ...], α (may) Case-mark β if

(i) α governs β

(ii) α is adjacent to β

(iii) α is [$-$N] (a potential Case assigner)

---

[41] The correct sequence of auxiliaries is maintained by the auxiliary system of Chapter 3. The optional *agent* role is assigned by the preposition *by* to its object. Linking of the *agent* role assigned in the *by*-phrase to the "suppressed" external θ-role of the verb is done in a stipulative and not fully explained fashion in GB theory.

Condition (105.ii) is invoked in the explanation of the grammaticality judgements in (106).[42]

(106) a. Paul *quickly* opened the door.

b. Paul opened the door *quickly*.

c. * Paul opened *quickly* the door.

It is important to observe, though, that under a strict interpretation of the X-Bar theory, (106.a-b) are possible D-Structures, while (106.c) is not. Adjuncts may be generated only at the double-bar level, so they may not intervene between a head and its complements. The Case filter is not necessary to mark ungrammatical (106.c) under the noted X-Bar assumptions. However, if the base component is extended so that adjuncts may be generated/inserted at the single-bar level, as discussed in section 2.3.5, the adjacency condition becomes again relevant to explain (106.c) and similar cases.

The adjacency condition (106.ii) as stated is too strong to hold in general (Chomsky, 1981); it yields a number of unwarranted predictions and hence must be supplemented with several mechanisms to get around it. First, verb particles are free to intervene between a verb and its object (107.a-c), ignoring the adjacency condition. A typical explanation of the verb-particle construction assumes that the verb and the particle form a single constituent V, which acts as the relevant (adjacent) Case assigner. Closely related to the verb-particle constructions are occurences of various time, place, and other adverbials, as in (107.d-e).

(107) a. We looked *up* the answer.

b. Someone put *on* a bagpipe record.

c. The anthem contains the lines: "China has brought *forth* a Mao Tse-Tung."

---

[42] The data and judgements are from (Stowell, 1981).

d. I brought *home* a pound of salami.

e. It is amazing how many men of ability saw *not* the need to volunteer their services.

Another instance of (apparent) violation of the adjacency condition (105) is found in the focus NP-shift constructions (108). See (Stowell, 1981) for discussion of examples (108.a-c). (108.d-e), thanks to Slava Katz, show in striking detail that the material intervening between the verb and the "shifted" noun phrase can be several constituents long, in fact longer than the noun phrase.

(108) a. Kevin gave $[_{NP} \in]_i$ to his mother [a new book]$_i$

b. Brian brought $[_{NP} \in]_i$ back to America [a priceless portrait of Napoleon]$_i$

c. I heard $[_{NP} \in]_i$ on the radio [the song you told me about]$_i$

d. The film, made at the time of Che Guevara, brought $[_{NP} \in]_i$ for the first time into the open [the message of the guerillas]$_i$

e. Would jazz have the slightless interest if its function were only to bring $[_{NP} \in]_i$ back to life, in its own way and hence, necessarily, with less force and purity [the very same musical emotions of European art]$_i$

Focus NP-shift constructions are seldom discussed in the GB literature, as noted in section 2.3.4. In the analysis of (108), the D- and S-Structure position of the nominal is adjacent to the verb, satisfying the adjacency condition on Case assignment at S-Structure. The nominal is displaced to the observed position by NP-shift, perhaps in the Phonetic Form component. However, this analysis seems highly artificial and has several unsatisfactory aspects. First are details of realization noted in section 2.3.4. Presumably the landing position is not present at D-Structure, so NP-shift is not structure preserving; it is a new transformation in the grammar, distinct from move-α and quantifier raising. The second problematic aspect is the specification of the conditions under which NP-shift may apply. We run the risk of making the adjacency condition (105.ii) vacuous, since when it is not met, as in (108), we may invoke NP-shift.

The Phonetic Form component has several transformations like NP-shift, affix-hopping, deletion, etc., which are not subjected to the same constraints of transformations that apply at the

syntactic levels, D-Structure, S-Structure and LF. PF transformations may perform quite arbitrary rewriting, while the representations they produce are not subject to the same constraints that representations at the syntactic levels are. Except for an unconstrained phrase-structure component, none of the GB subtheories has rules or principles that apply at PF (cf. van Riemsdijk and Williams, 1986). It does not seem desirable to attribute to the PF component older-style transformations like NP-shift.

Saving the adjacency condition (105.ii) finds further empirical difficulty when we consider languages other than English. In Spanish, manner adverbials are free to intervene between a verb and its object (Stowell, 1981). In (109.a), the Spanish sentence corresponding to (106.c), which was used to motivate the Case filter, turns out to be grammatical. Certain manner and place adverbials are also free to intervene between a verb and its object (109.b,c).[43] In German, whose verbal position is VP-final at D-Structure, the direct object typically appears in VP-initial position with, possibly, material intervening between it and the verb (109.d). In non-configurational languages, which typically lack prepositions or postpositions and use instead Case-marking particles, the adjacency condition (105.ii) cannot hold for more than one constituent. As for the English double-object construction, the more elaborate inherent Case marking scheme must be assumed.

(109) a. Pablo abrio *rapidamente* la puerta.

b. lei *esta mañana* las noticias.

c. lei *en el bus* las noticias.

d. Wir müssen den Mann *Hier* zurücklassen.

We will hence assume that Focus NP-shift constructions may be base-generated. As we have seen in the examples above, sentences may violate in several ways the adjacency condition (105.ii). We also relax the X-Bar conditions by which non-arguments may not occur under the single-bar projection, as discussed in section 2.3.4. The use of recursive binary rules in section 2.3.4 allows VP-internal scrambling without actually increasing the number of base rules

---

[43] Stowell (1981) claims that time and place adverbials are not free to intervene in Spanish.

required. The expense is only in a small elaboration of the attribution associated with the productions. Assignment of thematic roles under VP scrambling is handled in the same way as subcategorization in section 2.3.4. In particular, we assume a one-to-one correspondence between elements in the subcategorization frame *subcat* of an entry and thematic roles in the internal thematic grid Θ_I of the same entry; this is considered in section 2.7. Alternatively, in place of two attributes *subcat* and Θ_I we could also assume a single attribute *GF-Theta* which is associated with lexical entries and lists internal thematic roles with the grammatical function under which they are assigned (but cf. Williams, 1981, on the redundancy in these representations).

In place of the adjacency condition (105.ii) we assume the simpler version (110). The feature $\pm$*Heavy* is binary and marks, under conditions identical to those of NP-shift, those noun phrases which may undergo NP-shift.[44] The empirical content of the revised condition (110) depends on the unspecified conditions.

(110) Case adjacency condition:
If α Case-marks β, then α is adjacent to β or β is [+ *Heavy*]

2.6.4 Structural determination of empty categories

Let us now examine the possibility of determining the functional type of an empty category from conditions at the position in which it occurs. Government-binding distinguishes four types of empty category (of class NP): Wh-trace, NP-trace, PRO, and pro. The first two are instances of trace, and arise due to the operation of move-α; the latter two are base-generated and behave like empty pronominals.[45] The functional type of these categories is encoded by the features *anaphoric* and *pronominal*, as discussed in section 2.3.3.5 and shown in (47). Under the transformational hypothesis, we might think that the functional type of a trace may be determined by lexical features of the phrase that occupies the position of the trace at D-Structure. However, this diagnostic is not always applicable, since it assumes the moved phrase is overt, giving no prediction for empty operators. Also, it simply gives the wrong

---

[44] Specification of the conditions under which a nominal is [+ *Heavy*] raises the same empirical questions as the conditions under which the nominal may undergo NP-shift.

[45] Small "pro" does not occur in English. We ignore it henceforth.

prediction in cases of NP-movement followed by Wh-movement. In (111) the leftmost trace is *Wh*, while the second is NP-trace.

(111) [Who]$_i$ [$_{IP}$ [$_{NP}$ e]$_i$ seems [$_{IP}$ [$_{NP}$ e]$_i$ to love Mary]]

Chomsky (1982) determines the type of an empty category by refering to the argument status of its position, and to properties of its antecedent, if it has one.[46] Variables (instances of Wh-trace) are in argument position (A-position) and bound by an antecedent in non-argument position (A-Bar position). That is, variables are in A-position and A-Bar bound. Recall that the functional type of a variable is non-*anaphoric* and non-*pronominal* If the category is *anaphoric*, it must either be ungoverned and free, or bound by an antecedent in A-position (A-bound), by axiom A of Binding. Hence NP-trace is A-bound, and PRO are either free or A-bound. If the category is, furthermore, either free, or bound by an antecedent with independent thematic role (θ-bound), it is *pronominal*, otherwise it is non-*pronominal* Thus PRO is either free or θ-bound, and NP-trace is θ-Bar bound. This discussion is simplified and summarized in table (112). The instance of Wh-trace in C-specifier position, not considered above, is the last entry in (112). Note that C-specifier is a non-argument position.

(112)

	A-position	**A-bound**	**θ-bound**
Wh-trace (variable)	+	−	n.a.
NP-trace	+	+	−
PRO	+	+/ free	+/n.a.
Wh-trace (COMP)	−	−	n.a.

Notice that the three conditions A-position, A-bound, and θ-bound in (112) uniquely identify the functional type of the empty category. The analysis crucially relies on the type of antecedent, if any, that an empty category has. If we eliminate the two conditions that refer to the antecedent (A-bound and θ-bound), we obtain no distinction between variable, NP-trace, and PRO -- i.e., almost no determination of functional type.

Correa (1987) looks at a different set of conditions to determine functional type of an empty category, in particular government, Case, and thematic role. Trace is (properly) governed, while PRO is not. A variable is in A-position and Case-marked, while NP-trace is not Case-marked.

---

[46] The elements of thematic and binding theory assumed here are covered below, in the respective sections. Thematic theory defines whether a given position is an argument or non-argument position.

Wh-trace in COMP is in A-Bar position, by definition. We obtain the structural determination (113) of functional type, according to the new set of conditions.

(113)

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

An important feature of the analysis (113) is that it does not refer to the antecedent of an empty category to determine its functional type. This property is extremely useful for the interpretive evaluation of chains. It means that we may determine the type of a category without determining or hypothesizing in advance the chain to which the category belongs, or otherwise the nature of its local antecedent, if one is assumed. It is sufficient to look at the argument status of the position, and the conditions of government and Case-marking at that position -- all three purely structural matters, as we have seen. The analysis (113) is a crucial ingredient of the Chain rule formulated in Chapter 3 for the computation of syntactic chains at S-Structure.

2.7 Thematic Theory

The idea of using semantic roles for the identification of the relations between predicates and arguments in a sentence is found in (Fillmore, 1968) and (Gruber, 1976). The identification of predicate-argument relations is an important aspect of the semantic interpretation of a linguistic expression. Each predicate is said to be related to its arguments by some collection of "thematic relations" or roles (θ-roles) that the predicate assigns to the arguments. The notions of agent, patient, theme, goal, source, etc, are familiar from traditional grammar and used for names of θ-roles. While the notion of thematic relation is present in the work of Fillmore, Gruber, and earlier Extended Standard Theory (Chomsky, 1977), the substantive elements of thematic theory were not articulated until the GB theory appeared (Chomsky, 1981).

2.7.1 Predicates and arguments

The notions *predicate* and *argument* are notions which should be defined by linguistic theory. Consider (114). A standard logical translation of (114) identifies the verb *love* as the predicate of the expression and the phrases *John* and *Mary* as its arguments. Lexical properties of the verb indicate that it may accept two arguments. The order of the arguments is important. If the first

argument position is identified with the grammatical Subject, an *agent* role is assigned at it. The second argument position is identified with the Object and assigned a *theme* or *patient* role.

(114) John loves Mary.
love(*John*, *Mary*)

The identification of predicates and arguments in an expression is not a straightforward matter, as illustrated in (115) with the glosses provided for each sentence. Predicates may be "higher-order" and arguments are not limited to the category NP. In (115.a-b) the first argument is higher-order, since it applies to the second predicate, and one argument is a clause, with its own predicate and set of arguments. In (115.b) the clausal argument is, furthermore, discontinuous. In (115.c), the third argument is a prepositional phrase.[47] The arguments of a predicate need not be overt. In (115.d) the first argument does not have a counterpart in the surface expression. This argument would be realized as Subject in more typical declarative sentences. In (115.e), the second argument in the literal is also implicit; it is a logical variable bound by the interrogative operator *who*.

(115) a. John thinks Mary loves him.
think(*John*, love(*Mary*, *him*)), where *him* = *John* or someone else

b. John seems to love Mary.
seem(love(*John*, *Mary*))

c. The clerk filed the report in the cabinet.
file(*the clerk*, *the report*, *in the cabinet*)

d. It is not easy to understand this sentence.
For arbitrary person *x*, ¬ easy(understand(*x*, *this sentence*))
(where *' this sentence'* is to be substituted by the formula above, ad infinitum)

---

[47] It is sometimes assumed that prepositions are Case-markers with no semantic content and that the second argument of the verb in (115.c) is not the PP *in the cabinet*, but rather the NP *the cabinet*. We deviate from that view here and take the whole PP to be the complement of the verb.

e. Who does John love?
For which person x, love(*John*, x)

One parameter of a syntactic theory is to decide whether implicit semantic arguments are to be represented overtly in syntactic representations (trees), for example, by occurrences of empty categories. Government-binding theory chooses the explicit representation approach. Arguments missing in their canonical positions on the surface of utterances are represented explicitly by an empty category. In GPSG (Gazdar, et al., 1985), an argument missing from a constituent may appear implicitly as a feature (SLASH) in a derivation tree. Assuming the framework of augmented phrase-structure grammar of most current linguistic theories, including GB, the question of whether an argument should be represented overtly as a branching element in a phrase-marker or implicitly as a syntactic feature becomes, in my opinion, only a marginal issue. See (Jensen, 1988) and section 2.1.3.3 for more details on this.

In the glosses in (114) and (115) of the logical forms of the examples tense and aspect information is ignored. Place, time, and manner adverbials and prepositional phrases, which may appear optionally in the sentences, are also ignored. These are not treated syntactically as complements of the verbs, but rather as adjuncts, and their interpretations do not appear as arguments of predicates in logical form. Thematic theory does not apply to this class of constituents.

We do not develop a formal characterization of the notions *predicate* and *argument* in linguistic expressions, but only an intuitive one, elicited by the examples (114) and (115). These notions cannot be developed precisely in the absence of a logical framework, since they are, primarily, logical notions. The problem is explored in (Kornfilt and Correa, in preparation), where Jackendoff's (1983) system of conceptual structure is extended into a logical framework for the representation of the meaning of lexical items.

2.7.2 Thematic relations

The thematic relation between an argument and a predicate is identified by a descriptive label. This label indicates the thematic role that the argument plays in the situation denoted by the predicate. We assume here the thematic relation labels (116) proposed by (Gruber, 1976).

(116) *agent, theme, goal, source, place, ... situation*

Each predicate has associated with it a list Θ of the thematic relations that it assigns to its arguments. We identify the elements of this list with place-holders or argument positions in the logical form of the predicate.[48] Once the interpretation of the predicate is fixed, each place-holder variable is assigned a name from one of the labels (116), in some systematic manner. The name assigned defines the thematic relation that the variable bears to the logical form in which it occurs and has no theoretical interest. It is only a mnemonic aid. For the predicate *love*, with logical form (117.a), the names assigned to the variables *x* and *y* are *agent* and *theme*. The list Θ of place-holders that is obtained is (117.b), assuming the mnemonic names.

(117) a. LOVE(*x*, *y*)

b. love.Θ : (*agent*, *theme*)

In (117) we have underlined one of the thematic roles associated with the predicate. Government-binding assumes that among the thematic roles associated with a predicate at most one may be lexically marked as an *external argument*, to distinguish it from the other roles, which are then called *internal*. This elaboration of logical forms, which might seem like an unorthodox complication of the logical notion of "argument," has the motivation discussed in section 2.7.3 below on thematic role assignment.[49] The list of thematic relations associated with a predicate, together with an indication of which one is external, is called its *thematic grid* (or θ-grid). The underlining convention used to mark the external argument is due to Williams (1981). The θ-grids of the predicates in (115) appear in (118).

(118) a. think.Θ : (*agent*, *situation*)

b. seem.Θ : (*situation*)

c. file.Θ : (*agent*, *theme*, *place*)

d. be.Θ : (*situation*)

---

[48] More precisely, Θ is the list of free variables in the logical form of the predicate.

[49] In standard logical languages (e.g., the predicate calculus) there is not distinction between internal and external arguments of a predicate.

The use of thematic relations in Government-binding is significantly different from their use in other grammatical theories which also refer to them (Fillmore, 1968; Jackendoff, 1972). Chomsky (1981) hypothesizes that thematic relations are primitive notions with an epistemological status different from grammatical relations. Thematic relations are part of the primitive vocabulary in terms of which linguistic data are analyzed. In contrast, grammatical relations are structurally defined. Although thematic relations have clear (intuitive) semantic notions attached to them in the theories of Fillmore and Jackendoff, in GB theory no semantic significance is attached to them. Rather, thematic relations in Government-binding function only as "abstract syntactic relations holding between complements and heads" (Ravin, 1987).

Kornfilt and Correa (in preparation) propose a mechanism by which thematic relations are defined structurally on semantic representations (logical forms). This proposal brings together thematic and grammatical relations; both are defined structurally, although at different levels in the grammar. It contrasts with the current GB approach to θ-roles, since the new notion of θ-role is not a label that a predicate assigns to its argument, but rather a structural relation that the argument enters into with the predicate when its reading is substituted into the reading of the predicate. We believe this approach to θ-roles eliminates some of the problems noted by Ravin (1987) that arise from a primitive view of θ-roles. Thematic relations become non-primitive concepts. Under the new approach, there is an actual connection between the meaning of a lexical item (its conceptual structure) and its thematic grid, hence with its syntactic behavior. The distinction between the meaning of a predicate and its thematic grid is also made clear, making it possible to distinguish the meanings of items with identical θ-grids, such as *eat* and *kill*, and to identify the meaning relations between items related by synonymy or hypernymy, but with different θ-grids (cf. Ravin's discussion of *die*, *eat*, and *kill*).

Ravin's criticism of the impossibility to draw semantic relations (in Government-binding) between different lexical items is based on the view that a thematic grid is the only item of semantic information stored in the lexicon. But this is view is not valid. Traditionally, the lexicon has been regarded an association of form and meaning. (Chomsky, 1965) takes "a lexical entry to be simply a set of features, some syntactic, some phonological, some semantic." The weakness of thematic theory in Government-binding arises when attention to lexical information is restricted to the syntactic and phonological elements only, as in (Chomsky, 1965, 1981). Even though semantic representations are assumed in the lexicon as part of an item's entry, their relation to the syntax is not pursued, with some exceptions (Levin and Rappaport, 1986).

We believe the shortcoming of thematic theory may be traced to the fact that Government-binding grammar has not developed a notation in which the meaning of lexical items may be represented. Jackendoff (1983) addresses this question. Government-binding assumes θ-grids are independent features of lexical items, without pursuing their relation to semantic forms. The GB notion of θ-role provides no justification for the θ grid (117) of *love*, rather than, say, the implausible (*goal, source*). There are no principles by which the lexicon may be prevented from listing those thematic grids. The "structural" definition of θ-roles proposed by Kornfilt and Correa deals with this problem.

2.7.3 Thematic role assignment

The notions of predicate, argument, and thematic relation have been established informally above. Our next step concerns process of thematic role assignment or, equivalently, the identification of predicate-argument relations in linguistic expressions. The problem is not a simple one. An expression may have several predicates in it, each with its own set of arguments. The problem may be formulated as follows: Let S be a linguistic expression and let $P = \{ p_i(\ldots, \alpha_{i,j}, \ldots) : i, j \geq 0 \}$ be the set of predicates in S; $\alpha_{i,j}$ is the j-th argument position of predicate p_i. Let $A = \{ a_i : i \geq 0 \}$ be the set of arguments in S. At this time, we may assume that the problem of identifying arguments and predicates in S is solved by an independent analysis procedure. The predicate-argument structure problem is to devise the map (119) from arguments in S to argument-positions of predicates in S.[50]

(119) $\quad A \rightarrow \{ \alpha_{i,j} : i, j \geq 0 \}$

Government-binding uses the descriptive apparatus of the lexicon, base, and transformational components to define predicate-argument relations. The lexicon provides a subcategorization frame and thematic grid for each lexical item (which is a predicate). The subcategorization frame is the value of the *subcat* feature of the item. For notational convenience, we assume that a predicate's thematic grid Θ is encoded in two attributes Θ_i and θ_E, of internal and external

---

50 All argument positions of a predicate appear overtly in a syntactic representation, perhaps as empty categories. Due to this theory-internal assumption, the cardinalities of the sets of arguments and argument positions in the problem is identical. This follows from the Thematic criterion discussed below.

θ-roles, respectively. With this convention, the entry for the verb *file* might contain the feature specifications (120).

(120) **file**
subcat: [NP, PP]
θ_E : *agent* Θ_I: [*theme, place*]

Internal θ-roles are assigned to subcategorized complements of the predicate. There is a one-to-one correspondence between categories in the subcategorization frame and θ-roles in the internal thematic grid Θ_I. The correspondence is represented by the order in which subcategorized categories and internal θ-roles are listed. If θ_i is the *i*-th internal θ-role and XP_i the *i*-th subcategorized position, the grammatical relation in which θ_i is assigned is [XP_i, VB].[51]

The external θ-role of a predicate is assigned to the Subject position associated with the predicate. In (Chomsky, 1981), the external θ-role is assigned by VP, which synthesizes the θ_E attribute of its head. Chomsky actually entertains the possibility that the external θ-role is determined compositionally by the verb and its complements, but the details of the composition are not elaborated. Here we let the external thematic role θ_E be synthesized by VP, as a copy of the value of same feature on the head verb. Since the phrase structure rules we adopt do not have VP as sister of Subject position, it is necessary to assume that the θ_E feature is further synthesized by the parent IB node. It is this node which assigns the external θ-role to the Subject. A sample sentence with the relevant phrase structure is shown in (121).

(121) $[_{IP}$ [John] $[_{IB}$ $[_{VP}$ refused the offer]]]

In derived nominals, the noun may have thematic and subcategorization features similar to those of the original verb. Due to the asymmetry between IP and NP, the assignment of the external θ-role in NP proceeds differently from that in IP. In (122), the noun *refusal* assigns its external θ-role to the Subject *John* via the NB node.

(122) $[_{NP}$ [John's] $[_{NB}$ refusal of the offer]]

---

[51] Williams (1981) presents a mechanism for the derivation of the subcategorization frame of a predicate from the predicate's thematic grid. Kornfilt and Correa (in preparation) derive both subcategorization frames and θ-grids from conceptual representations.

In this discussion we have not identified the level (or levels) of representation at which θ-role assignment takes place. Thematic relations are assigned at D-Structure, as described above. A syntactic position at which a θ-role may be assigned is called an *A-position* (argument position). A-positions are identified by the grammatical functions Subject-of-X and complement-of-X, where X is N, V, P, or A. All other positions, in particular the places of adjuncts and complementizers, are called *A-Bar positions* (non-argument positions). A position at which a thematic role is assigned is called a θ-position. By the one-to-one correspondence between (subcategorized) complement positions of a predicate and the elements of the internal thematic grid of the predicate, every complement position is a θ-position. Subject position, on the other hand, may be a non-θ position.

A constituent occupying an argument position "inherits" the θ-role assigned at that position, if any, and is thus identified as an *argument* of the predicate that assigned the θ-role. A sight elaboration of the foregoing is required. If the constituent in question is an operator, such as a quantified or wh-phrase, then the operator does not occur at the position at LF and a logical variable is hypothesized instead. The variable is then bound by the operator. It is the variable which becomes identified as an argument of the predicate. In (115.c), repeated here as (123.a), the constituents "the clerk", "the report", and "in the cabinet" are identified, respectively, as the *agent*, *theme*, and *place* arguments of the predicate *file*.

(123) a. [the clerk] filed [the report] [in the cabinet].

b. [the report]$_i$ was filed [$_{NP}$ e]$_i$ [in the cabinet] (by [the clerk]).

c. [what]$_i$ do you think [$_{CP}$ [$_{NP}$ e]$_i$ [the clerk] filed [$_{NP}$ e]$_i$ [in the cabinet]].

Move-α may displace a constituent α_0 from its D-Structure position, to some other position at S-Structure. If $(\alpha_0, ..., \alpha_n)$ is a chain, then α_0 "inherits" the θ-roles assigned at the positions α_i in the chain, for $i = 0, ..., n$. In (123.b), the argument "the report" is assigned the *theme* role, since its chain ([the report]$_i$, [$_{NP}$ e] $_i$) is assigned that role at the last position. Notice that no θ-role is assigned to the head of the chain, in Subject position, since the passive form of the verb assigns no external θ-role. Hence *theme* is the only θ-role assigned to "the report." The suppressed *agent* θ-role remains available, in some unspecified way, to be assigned to the object of the *by*-phrase, if one is present. Similarly, in (123.c), the phrase "what" is linked to the *theme* role by

the same mechanism. Since the phrase is an operator, the trace in abject position is a logical variable bound by the operator.

2.7.4 Thematic criterion and Projection principle

Parallel to the Case filter (99) of Case theory, thematic theory imposes the thematic criterion (θ-Criterion) (124) on the distribution of thematic relations (Chomsky, 1981).

(124) Thematic criterion:

(i) Each argument is assigned one and only one θ-role, and

(ii) each θ-role is assigned to one and only one argument.

As with the Case filter, it is necessary to indicate the level (or levels) at which the θ-Criterion is intended to apply. The Projection Principle (125) states this; this principle, supplemented with the condition that all clauses have a Subject position, is known as the Extended Projection Principle (Chomsky, 1981).

(125) Projection Principle:
The thematic criterion holds at D-Structure, S-Structure, and Logical Form.

Since thematic relations are assigned at D-Structure, the θ-Criterion implies that at D-Structure the mapping between arguments and argument positions in an expression is a bijection. The Projection Principle requires further that the D-Structure bijection be maintained at all other syntactic levels of representation, even if syntactic movement of arguments takes place.[52]

The θ-criterion and projection principle are involved in the explanation of the grammaticality judgements in (126). In (126.a), the predicate *see* assigns roles *agent* and *theme* to its two arguments. In (126.b), though, there are multiple problems. At D-Structure, the D-object *John*

---

[52] Strictly speaking, the Projection Principle (11) does not require that the D-Structure bijection be maintained at S-Structure and LF, but rather the weaker condition that *a* bijection be maintained at these other levels, not necessarily the same one. But the intent of (125) is preservation of the D-Structure bijection.

is identified as *theme* and the *agent* role is not assigned, since there is no argument in D-Subject position. Hence the θ-Criterion is violated at D-Structure. By the Chain convention above, when movement takes place in (12.b) the single argument *John* is assigned both θ-roles, in violation of the θ-Criterion at S-Structure and LF. In (12.c), where the verb is passive, the external θ-role is suppressed, and neither of the previous two violations of the θ-Criterion happen; the single argument *John* is identified as *theme*. The remaining four examples in (126) may be analyzed similarly.

(126) a. John saw Mary.

b. * John_i saw $[_{NP} \in]_i$

c. John_i was seen $[_{NP} \in]_i$

d. * John was seen Mary.

e. It seems that John saw Mary.

f. * Bill seems that John saw Mary.

g. John_i seems $[_{NP} \in]_i$ to have seen Mary.

An important consequence of the Projection Principle is that movement of an argument is possible to either a non-argument position, such as complementizer or adjunct, or to a non-θ (argument) position. Movement to a θ-position leads to a violation of the θ-Criterion. This property of movement is used in the Chain rule of Chapter 3.

The θ-Criterion and Projection principle yield fine-grained distinctions about predicate-argument relations. This is done at the expense of the mechanism for θ-role assignment and very detailed lexical information in the lexicon. The lexicon is required to provide all possible thematic variants of each predicate.[53] Consider examples (127), bearing on the elaboration of the lexicon

---

[53] The lexicon may employ lexical rules to derive thematically related variants of a given predicate. The complication of the lexicon is not necessarily in terms of its size, but rather in terms of its internal structure, which now has to be more articulate (Levin, 1985; Hale and Kayser, 1986).

due to the introduction of thematic theory. The verb *break* has the transitive and intransitive uses shown. Levin (1985) refers to this transitivity alternation as the anti-causative alternation and suggests that it may be predicted by the "semantic class" to which the verb belongs. In the transitive use, the verb selects two arguments and is causative. The θ-grid defined in the lexicon is (*agent*, *theme*), where the two roles are assigned at Subject and object positions, respectively. In the intransitive use (127.b), the verb selects a single argument *theme* and is non-causative. The lexicon must provide a different θ-grid for this alternate form. At this point, there is a choice for selecting the θ-grid of the predicate: Should it be (*theme*), or perhaps (*theme*)? GB theory does not provide a criterion to chose between these two alternatives. Either may be used in the derivation of a valid S-Structure, as in (128). (128.a) involves a simpler derivation, but Levin (1985) argues that (128.b) should be preferred.

(127) a. John broke the window.

b. The window broke.

(128) a. [The window] broke.

b. $[\text{ The window}]_i$ broke $[_{NP}\, e]_i$

2.8 Binding Theory

Noun phrases in a sentence may have a common reference. The objective of the Binding theory is to identify the conditions on coreference that exist between any pair of noun phrases occurring in the same sentence. The theory indirectly determines constraints on the distribution of certain kinds of noun phrases and thus contributes to the overall notion of grammaticality defined by the grammar. In this section we discuss the formulation of the Binding theory, as presented in (Chomsky, 1981, 1986a).

Before we proceed, it should be remarked that the Binding theory underdetermines the constraints on referencial possibilities of noun phrases. The principles proposed by the Binding theory are clause-bound syntactic factors which must be supplemented by many extra-syntactic factors and heuristics to achieve a full theory of anaphora. The reader is referred to (Hobbs, 1978) for discussion of some of the extra factors that might be involved.

2.8.1 Referential indices

The phrase from which a noun phrase takes its reference is called its *antecedent.* We denote coreference between two phrases by subscripting an identical referential index *RefIndex* to the phrases, as in (129). The second index of each pair subscripted in (129) is our referential index. Two noun phrases are said to be *coreferential* if they bear the same referential index. *RefIndex* is an integer-valued syntactic feature on the on the root of a phrase, distinct from the *Chain* index introduced in section 2.2 as part of the rule move-α. When subcripted to a nominal, a referential index looks identical to the trace index *Chain*; however, the use of the two indices *RefIndex* and *Chain* should be kept separate.[54]

(129) $\text{Mary}_{i,k}$ seems $[_{CP}\ [_{NP}\ e]_{i,j}$ to like $\text{herself}_{h,j}\ ]$

2.8.2 Nominal types and referential possibilities

The referential possibilities of a noun phrase depend on the functional *type* of the noun phrase and certain stated Binding conditions for that type. For the purposes of the Binding theory, Government-binding distinguishes three types of noun phrase, as in (130).

(130) a. anaphors (reflexives and reciprocals)

b. pronominals

c. referential expressions (or names)

The type of a nominal is encoded by the binary syntactic features *anaphoric* and *pronominal*, as discussed in sections 2.3.2 and 2.3.3 and summarized in table (47). The type does not depend on whether the nominal is empty or not. If the nominal is overt, its type is synthesized from the type of its head, while if the nominal is empty the type is determined structurally as discussed in section 2.6 on Case theory.

---

[54] If the NP is a trace, we assume its referential index is identical to the trace index, which is determined by move-α. Thus trace indices are also subject to the binding axioms. We discuss this in more detail in Chapter 3.

2.8.2.1 Anaphors

An anaphor is an expression that has no independent reference and must take it from some other phrase in the sentence. English has *reflexive* and *reciprocal* anaphors, such as *themselves* and *each other* in (131). Since the anaphor must have an antecedent within the sentence in which it is used, we obtain the contrast between (131.a) and (131.b). If there is no appropriate antecedent, the string is ill-formed. The antecedent of the anaphor must, furthermore, preceed and c-command the anaphor,[55] and be found within a certain *local domain*, the latter notion to be made precise below. Thus, in (131.c), although there is a potential antecedent for the anaphor, namely *Greeks*, it is not within the required local domain. In (131.d), there is also a potential antecedent *donkey*, but it does not c-command the anaphor. The string is also ill-formed.

(131) a. Greeks$_i$ like *each other/themselves*$_i$

b. * *Each other/Themselves* like Greeks.

c. * Greeks$_i$ think that *each other*$_i$/*themselves*$_i$ are smart.

d. * Every man who owns a donkey$_i$ beats *itself*$_i$.

2.8.2.2 Pronominals

A pronominal is a pronoun in any of its inflected forms (e.g., as due to agreement and Case-marking) (132). Pronominals exhibit a distribution in phrase structure trees nearly complementary to that of anaphors. A pronominal need not pick its reference from some other phrase in the sentence, but rather may have independent (deictic) interpretation, as in the first reading of (132.a). The pronominal may also be read anaphorically, having its reference determined by some other phrase in the sentence (132.a,b). In this case, though, the antecedent must either be outside the local domain of the pronominal, or not c-command it. Hence, the assigned coreference in (132.a,b) is possible, while that in (132.c) is not. Within a local domain, where an anaphor must have an antecedent, a pronominal may not.

---

[55] This should put in perspective Langacker's Pronominalization condition, discussed in section 2.5.1.

(132) a. Brigitte$_i$ said that *she*$_{j/i}$ is tired.

b. Every man who owns a donkey$_i$ beats *it*$_i$.

c. * Sibylle$_i$ loves *her*$_i$.

2.8.2.3 Referential expressions

Lexical or fully referential expressions are names, like "John" and "the man" in (133); the class includes all nominals headed by a common or proper noun. A referential expression may determine its referent independently; it must be free in every domain, in the sense that it may not have a c-commanding antecedent. Thus the interpretations in (133.a,b) are unwarranted. Coreference between two referential expressions is possible only if neither c-commands the other (133.c,d). The result of coreference, though, may be awkward or place emphasis on the anaphoric expression (133.c-d).

(133) a. * John$_i$ likes *John*$_i$.

b. * John$_i$ wants that *John*$_i$ leaves.

c. The man who hired John$_i$ likes *John*$_i$.

d. John$_i$ came and *John*$_i$ left.

2.8.3 Local domains

The most difficult area of the Binding theory is the formulation of the notion *local domain* referred to above. The intuitive notion is that each nominal has associated with it certain syntactic environment, within which its coreference conditions or *binding axioms* must be satisfied. We adopt here the formulation (134) of local domain (Chomsky, 1981). The notion defined is purely structural and does not depend on the functional type of the nominal in question. The notion of minimal governing category is (83), as defined in section 2.5.3 above. In spite of the problems that we will note in regard to (136) below, we assume, contrary to (Huang, 1982), that the SUBJECT-accessibility requirement in (83.iii) applies to both anaphors and pronominals.

(134) **Local domain:**

The *local domain* of a nominal α is the subtree dominated by MGC(α), the minimal governing category of α.

By definition (83), β is a minimal governing category of α if β dominates α, a governor of α, and a SUBJECT accessible to α (Chomsky, 1981). The term SUBJECT in the definition has the sense (i) Subject (NP) of an infinitival sentence, (ii) Subject of a noun phrase, (iii) Subject of a small clause, and (iv) the *AGR* element in a tensed sentence. Ignoring sense (iii) of SUBJECT, IP and NP are the only two categories which may have a SUBJECT; it follows that IP and NP are the only nodes that are potential minimal governing categories.

Analysis of the notion MGC and definition (134) against the examples in (135) reveals the following. In (135.a) the MGC of *themselves* is the IP node marked; hence the anaphor's local domain is the entire clause, and the anaphor finds the marked antecedent therein. Similarly, in (135.b) the MGC of *themselves* is the IP node, and we have the same situation as before. The object NP is not a MGC since it lacks a specifier (SUBJECT). In contrast, in (135.c) the object NP has a SUBJECT and counts as the MGC of the anaphor. Consequently, the anaphor may not take its antecedent outside the local domain defined by the NP. It is shown in (135.d) that the NP specifier may serve as antecedent for the anaphor. In (135.e), since the complement clause is tensed, the embedded *AGR* element is an accessible SUBJECT for the anaphor *themselves* and the IP node marked is the MGC of the anaphor. The marked coreference relation is not licit, since the antecedent is outside the local domain defined by the complement IP. In contrast, in (135.e) the complement clause is untensed, and thus does not contain a governor for the anaphor *themselves*. It follows that the MGC of the anaphor is the matrix IP node, and the local domain is the entire clause. The marked coreference relation thus obtains.

(135) a. $[_{IP}$ Greeks$_i$ like *themselves*$_i$ $]$

b. $[_{IP}$ Greeks$_i$ like $[_{NP}$ stories about *themselves*$_i$ $]]$

c. * $[_{IP}$ Greeks$_i$ like $[_{NP}$ Homer's stories about *themselves*$_i$ $]]$

d. $[_{IP}$ Greeks like $[_{NP}$ Homer's$_i$ stories about *himself*$_i$ $]]$

e. * Greeks$_i$ believe that [$_{IP}$ *themselves*$_i$ are smart]

f. [$_{IP}$ Greeks$_i$ believe [$_{IP}$ *themselves*$_i$ to be smart]]

The notion local domain defined in (134) is identical for pronominals and anaphors; it does not depend on the functional type of the nominal to which it applies. The binding axioms to be formulated below predict that in the domain where an anaphor must have an antecedent, a pronominal cannot. This gives rise to the problem noted in (136), where the local domain of the italicized nominals is the object NP.

(136) a. The children$_i$ like *each other's*$_i$ pictures.

b. The children$_i$ like *their*$_i$ pictures.

c. * The children$_i$ like *themselves'*$_i$ pictures.

The nominals in (136.a-b) are of different types, anaphor in (136.a) and pronominal in (136.b), and yet they may refer back to the same expression; this contradicts the binding axioms below. Note that this problem disappears if we assume as (Huang, 1982) that SUBJECT-accessibility applies to anaphors and not to pronouns. It is clear that (134) needs refinement so that it distinguishes between anaphors and pronominals, and also between the two kinds of anaphors, reflexives and reciprocals. In (136.c) the reflexive NP cannot refer to the Subject as antecedent, in contrast with (136.a). The notion local domain, although close for reflexives, pronominals, and reciprocals, is probably not the same, as illustrated in (136) (Chomsky, 1986). In this thesis we do not pursue refinement of definition (134); this is an open problem in Government-binding theory.

2.8.4 Free indexing

Having reviewed informally the conditions on coreference of nominals and defined the notion local domain, we proceed to discuss Chomsky's axiomatic statement of the Binding theory. Chomsky (1981) assumes the *Free indexing rule* (137), which applies in the derivation of Logical Form and assigns (randomly) a value to the referential index of each NP in the input structure.

(137) **Free indexing rule:**
Assign (referential) indices freely.

The proposed indexing rule massively overgenerates logical forms; It assigns indiscriminately unwarranted coreference relations. The annotated structures produced by the rule are subject to a number of well-formedness conditions that we now discuss; the said conditions filter out unwarranted interpretations. In Chapter 3 we develop an alternative to the Free indexing rule (137), incorporating the binding axioms into the operation of the rule, in such way that it may be guaranteed to generate only well-formed LF structures.

2.8.5 Binding conditions

We begin assuming that every NP node has been assigned a referential index by the Free indexing rule (137). The most elementary well-formedness condition is the agreement condition (138). The main component of the theory consists of the Binding axioms (139), where the notion of *binding* is defined in (140). The definition of local domain is (134) above and has the problems noted there. The notion of c-command in (139) is that of definition (72). Condition (138) and the Binding axioms (139) apply at LF, after the Free indexing rule has applied.

(138) **Agreement Condition**
If NP_1 and NP_2 are coindexed, then their agreement features
$NP_{1,2}.AGR = <Person, Gender, Number>$ coincide.

(139) **Binding Axioms**

A. An *anaphor* must be bound within its local domain.

B. A *pronominal* must be free within its local domain.

C. A *referential expression* must be free in every domain.

(140) For nodes α and β, α *binds* β if (i) α is coindexed with β, and (ii) α c-commands β. A node α is *free* (within a given domain) if it is not bound (within that domain).

The agreement condition (138) between the *AGR* features of two nominals blocks unwarranted interpretations, as those in (141).

(141) a. * She$_i$ washed himself$_i$.

b. * John$_i$ wants that she$_i$ wins the race.

Notice that a given node α may be coindexed with a node β and be identified correferent with it, while it does not follow that α binds β, in the sense of definition (140). An instance of his situation is (132.b), where neither of the coindexed nominals c-commands the other. Notice also that the binding relation need not satisfy subjacency (84), since more than one bounding node, IP and NP, can occur in the path connecting the two nodes coindexed. This can be seen by deeper embedding of the coindexed pronoun in (141.b).

It is a straightforward task to verify that the Binding axioms in (139) explain the grammaticality judgements and interpretation possibilities of the examples presented thus far, except those in (136). The theory emboddied in definitions (134) through (140) is not particular to English, and covers a wide domain of empirical data, as shown in the examples. These definitions constitute a good example of the degree of descriptive and explanatory depth of Government-binding grammar.

2.9 Bounding Theory and Movement

The formulation of the movement rule move-α in (5) and (16) above is such that it permits movement of a phrase [$_{NP}$ X] over an arbitrarily long distance in the input phrase-marker. Bounding theory imposes the subjacency relation defined in (84) as a constraint on the structural description of the transformation. The Subjacency condition as well as several other miscelaneous conditions on transformations are discussed in this section.

2.9.1 Subjacency condition

The "distance" between the target and the extraction sites in a phrase-marker applied move-α must be subjacent. The statement (16) of rule move-α must be refined as in (142), to include the Subjacency condition (142.c).

(142) a. **move-α:**

$W\ [_{\alpha}\ e\ ]\ X\ [_{\alpha}\ Y\ ]\ Z \rightarrow W\ [_{\alpha}\ Y\ ]\ X\ [_{\alpha}\ e\ ]\ Z$

b. **Trace convention:**

$\alpha_4.Chain \leftarrow \alpha_2.Chain$

$\alpha_3.Chain \leftarrow \alpha_4.Chain$

c. **Subjacency condition:**

subjacent(α_1, α_2)

The Subjacency relation, defined in (84), is repeated for convenience as (143). Notice that the Subjacency condition (142.c) is not an attribute condition, but rather a structural condition on acceptable factorizations of the input phrase-marker in (142.a); it does not refer to any of the attributes that may be associated with the nodes there. The formal status of (142.c) in the theory grammar is different from that of, say, the Case filter, which is a true attribute condition. It is of interest to see if (142.c) may be reduced to (or made a consequence of) independent conditions in the grammar. In Chapter 3 we show that the Chain rule we formulate satisfies the Subjacency condition, without need of stipulating it separately in the grammar.

(143) a. α is *subjacent* to β if and only if the path (X, ..., β), excluding the end nodes X and β, contains at most one bounding node; X is the highest node of the path (α, ..., β).

b. IP and NP are *bounding nodes* (for English).

We now illustrate the manner in which (142.c) enters into the explanation of grammaticality judgements in the grammar. Given that the phrase structure component is fairly unconstrained,[56] there are often several derivation paths in the derivation of a string. For the derivation of sentence (144.a), we assume we start with either of D-Structures (144.b) or (144.c). Assuming (144.b), a single application of move-α may move the D-Structure phrase *who* to its surface position, to derive S-Structure (144.d). Assuming (144.c), two derivation paths are possible. First, S-Structure (144.e) may be produced by a single application of move-α. Alternatively, S-Structure (144.f) may be produced by cyclic application of move-α, to move *who* to the

---

[56] This translates into overgeneration and ambiguity of the analyses it provides, as we show.

intermediate C-specifier position in the first cycle, and to the matrix C-specifier position in the second application. Notice that the only difference between (144.e) and (144.f) is in the intermediate empty category in C-specifier position; this category is not part of the chain formed by movement in (144.e), but it is in (144.f).

(144) a. Who do you think will serve this country best ?

b. $[_{NP} \in]$ $[_{IP}$ you do think $[_{IP}$ [who] will serve this country best]]

c. $[_{NP} \in]$ $[_{IP}$ you do think $[_{NP} \in]$ $[_{IP}$ [who] will serve this country best]]

d. $[\text{Who}]_i$ do $[_{IP}$ you think $[_{IP}$ $[_{NP} \in]_i$ will serve this country best]]

e. $[\text{Who}]_i$ do $[_{IP}$ you think $[_{NP} \in]$ $[_{IP}$ $[_{NP} \in]_i$ will serve this country best]]

f. $[\text{Who}]_i$ do $[_{IP}$ you think $[_{NP} \in]_i$ $[_{IP}$ $[_{NP} \in]_i$ will serve this country best]]

The three-way ambiguity that this aspect of the derivation alone presents disappears once the Subjacency condition (142.c) is introduced to constrain the application of move-α.[57] In the derivation of (144.d) and (144.e) illicit use of move-α is made. The structural descriptions input to move-α do not satisfy subjacency; there are two bounding nodes IP on the path connecting the source and target nodes of the movement. The Subjacency condition is also involved in the explanation of the ill-formedness of (145.a-b), whose D-Structure is parallel to (145.c). In this case, the Subjacency violation is due to the two bounding nodes IP and NP that intervene between the source and target nodes for movement.

(145) a. * Who_i did $[_{IP}$ $[_{NP}$ a book about $[_{NP} \in]_i$] appear last week]

b. * $[\text{About whom}]_i$ did $[_{IP}$ $[_{NP}$ a book $[_{PP} \in]_i$] appear last week]

c. $[_{IP}$ $[_{NP}$ A book about Nixon] appeared last week]

---

[57] Although we have stated the Subjacency condition as a constraint on the application of move-α, the condition may equally well be taken as a condition on the representations produced by the transformation, or the phrase structure component in general. See also (Lasnik and Saito, 1984).

The contrast that the choice of Bounding nodes brings about in the Subjacency condition may also be illustrated with the examples (145). The choice (143.b) of bounding nodes is a parameter of universal grammar, which may vary from language to language. For English the choice is the set IP and NP, as in (143.b); for Spanish and Italian the choice is argued to be CP and NP (Rizzi, 1982). Interestingly, the translation (146) of (145.b) is grammatical in Spanish. Assuming the new set of bounding nodes, CP and NP, no violation of the Subjacency condition obtains, since only one bounding node, NP, intervenes between the moved phrase and the extraction site.

(146) $[_{CP}[\text{Acerca de quien}]_i$ fue que $[_{IP}$ aparecio $[_{NP}$ un libro $[_{PP} \in]_i$ la semana pasada]]

The statement (142) of move-α is now a more accurate formalization of its definition in Government-binding theory. It incorporates the trace convention and the Subjacency condition, and pressuposes a mechanism for the initial evaluation of *Chain* indices at D-Structure. The rule illustrates well the rich set of formal grammatical devices assumed in Government-binding theory. (142.a), the original rule move-α, is a schema for an infinite collection of generalized rewriting rules. (142.b), the second component, is a pair of attribution rules that define the value of one of the attributes (*Chain*) assumed in the theory. Lastly, the Subjacency condition (142.c) is a condition on analyzability of the input phrase-marker, and hence part of the structural description of the transformation.[58]

2.9.2 Miscelaneous conditions on representations

There are several miscelaneous constraints on the representations produced by move-α, which do not follow from the revised definition (142) of the rule. These constraints are fairly idiosynchratic, and there has been significant attempt to reduce them to more general principles, such as the Subjacency condition, the ECP, discussed below, or the Binding axioms in section 2.8. In this section we will discuss the first constraint, the doubly-filled-COMP filter, and simply list the rest without commentary. In the following section we return to the *that*-trace filter, and

---

[58] Rule move-α is not defined as in (142) in the Government-binding literature. It may be argued that the rich set of grammatical devices in (142) should not be imputed to move-α, but rather to this particular definition. However, (142) is an accurate symbolic characterization of (part of) the rule verbally described in the literature.

in Chapter 3 we will have occasion to discuss the relation of the Path Containment Condition given below to the Subjacency condition.

First consider the phenomenon of doubly-filled complementizers, as in (147) (van Riemsdijk and Williams, 1986). Although the English complementizer provides positions for two constituents, a slot for Wh-phrases, and a slot for complementizers, notice that only one of these positions may be lexically filled at one time, from the observed S-Structures.

(147) a. I saw the man $[_C \phi_i$ -] you talked to $[\in]_i$

b. I saw the man $[_C \phi_i$ that] you talked to $[\in]_i$

c. I saw the man $[_C$ who$_i$ -] you talked to $[\in]_i$

d. * I saw the man $[_C$ who$_i$ that] you talked to $[\in]_i$

The doubly-filled-COMP filter (148) has been proposed to account for these cases. This condition is stated as a constraint on S-Structures, filtering out those that meet the structural description.[59]

(148) **Doubly-filled COMP filter:**

* $[_C \alpha \beta]$, where α and β have phonological content

The analysis (148) assumes that C has two optional constituents, according to the older X-Bar analysis of (Chomsky, 1981). In the analysis adopted here, the structure of CP is (149). The left position α is the C-specifier and the second, β, is dominated by C. Hence, the formulation of the doubly-filled COMP filter (148) must be modified to reflect this change in phrase structure.

(149) $[_{CP} \alpha [_{CB} [_C \beta] \text{ IP }]]$

---

[59] The filter is not stated as a condition on the structural description of move-Wh, in order to maintain the latter as an instance of move-α, which, except for the Subjacency condition, does not have conditions like (148).

The next three conditions, (150)-(152), are also constraints on S-Structure, much like the doubly-filled-COMP filter.

(150) ***that*-trace filter:**
* [$_{CP}$ *that* [$_{NP}$ e] ...], unless CP is in the context [N __]

(151) **C-command constraint on movement:**
A trace must be c-commanded by its antecedent.

(152) **Path containment condition:**
Let $A=(\alpha_1, ..., \alpha_n)$ and $B=(\beta_1, ..., \beta_k)$ be chain paths. Then either A and B are disjoint or one contains the other.

2.10 Empty Category Principle and Government

The last element of Government-binding grammar that we consider is the Empty Category Principle (ECP). This principle is the result of a long line of research started in the late seventies, attempting to unify several disparate constraints on the distribution of empty categories.[60] Some of these constraints, like the *that*-trace filter, were noted in the previous section.

The empirical domain of data that the ECP attempts to explain concerns the distribution of non-pronominal empty categories (NP-trace and Wh-trace) in Subject positions, as we proceed to illustrate. The data concerning the ECP is highly peripheral and controversial, as the grammaticality judgements it requires to analyze the data are quite subtle, even for native speakers. The data presented in this section are borrowed from the literature cited, and the more marginal judgements expressed do not reflect necessarily my own. First, consider the *that*-trace phenomena in (153) (van Riemsdijk and Williams, 1986).

---

[60] The reader should not get the impression that the ECP is firmly established in GB grammar. Its formulation suffers from a number of shortcomings and is rather controversial; there are recent attempts to reduce the ECP to yet other sets of principles (Aoun, 1985).

(153) a. Who$_i$ do you think [$_{CP}$ [$_{NP}$ ∈]$_i$ [$_C$ ∈] Bill saw [$_{NP}$ ∈]$_i$]

b. Who$_i$ do you think [$_{CP}$ [$_{NP}$ ∈]$_i$ that Bill saw [$_{NP}$ ∈]$_i$]

c. Who$_i$ do you think [$_{CP}$ [$_{NP}$ ∈]$_i$ [$_C$ ∈] [$_{NP}$ ∈]$_i$ saw Bill]

d. * Who$_i$ do you think [$_{CP}$ [$_{NP}$ ∈]$_i$ that [$_{NP}$ ∈]$_i$ saw Bill]

The phenomenon in (153) is accounted for in (Chomsky and Lasnik, 1977) by the (simplified) *that*-trace filter (150) above. Again, as in the case of the doubly-filled-COMP filter, (150) is a constraint on S-Structures, rather than on the rule move-α itself. Assuming the analysis of (Chomsky, 1981), a crucial property of example (153.d) is that the trace in complementizer position does not c-command the trace in Subject position, since the C node branches by dominating the two constituents shown. The definition of c-command is not used strictly in (153.c); it is assumed that the node C in (153.c) does not count as a branching node, since there is no overt complementizer. The *that*-trace filter is a stipulative constraint reduced to the ECP below.

The second class of phenomena that the ECP is argued to explain are the *parasitic gap* constructions in (154) (Chomsky, 1982). The characteristic that defines these cases is the occurence of two traces, bound by the same operator in C-specifier position.

(154) a. * Someone who$_i$ John expected [$_{CP}$ [$_{NP}$ ∈]$_i$ would be successful though believing [$_{CP}$ [$_{NP}$ ∈] is incompetent]]

b. Someone who$_i$ John expected [$_{CP}$ [$_{NP}$ ∈]$_i$ to be successful though believing [$_{CP}$ [$_{NP}$ ∈] to be incompetent]]

At work in the examples is the assumption that in (154.b), but not (154.a), the parasitic gap (second trace) is properly governed by the *ecm* verb preceeding it.

The Empty Category Principle is stated in (155). Its substance depends on the definition of proper government in (82), repeated here as (156) for convenience. This definition is taken from (Chomsky, 1981). The class of governors for the ECP, as follows from the definition of proper government, includes the (zero-level) lexical categories X^0, as well as maximal projections XP_i

bearing a trace index. The non-lexical I, even if bearing a non-*nil AGR* marker, does not count as a proper governor. As remarked above, the ECP applies only to the non-pronominal ([-*pronominal*]) empty categories, namely NP-trace and Wh-trace, but not PRO or pro .

(155) **Empty Category Principle (ECP):**
$[_{NP} \in]$ must be properly governed.

(156) α *properly governs* β if and only if

(i) $\alpha = X^0$ or is trace-coindexed with β, and

(ii) α c-commands β, and

(iii) Each maximal projection ϕ which dominates β also dominates α, and

(iv) $\alpha \neq AGR$ (we read $\alpha \neq$ I)

Definition (156) of proper government assumes the X-Bar phrase-structure of (Chomsky, 1981). The ECP then subsumes the *that*-trace filter (150), as shown in (157), a more detailed rendering of (153.c-d). Crucially, the C node in (157.b) is branching, dominating both the trace $[_{NP} \in]_i$ and the complementizer *that*. Hence the trace in C does not c-command the trace in Subject position, and does not govern or properly govern it. Since I is neither a proper governor for the latter trace (although I governs it), the Subject trace is not properly governed, in violation of the ECP. In contrast, in (157.a) the C node is not branching, and the trace in C properly antecedent-governs the trace in Subject position. The trace in C, in turn, is properly governed by the preceeding verb, by yet another extension of government: If α governs β, then α also governs the head and specifier of β (see Chomsky, 1982).

(157) a. Who$_i$ do you think $[_{CP}\ [_{NP} \in]_i\ [_{IP}\ [_{NP} \in]_i$ saw Bill]]

b. * Who$_i$ do you think $[_{CP}\ [_{C}\ [_{NP} \in]_i$ that] $[_{IP}\ [_{NP} \in]_i$ saw Bill]]

The definition (156) of proper government is rendered completely useless under the X-Bar phrase structure analysis of IP and CP in (Chomsky, 1986b), adopted here. Under that analysis, the position for Wh-movement is C-specifier position. This position c-commands IP and IP-Subject

position, regardless of whether there is an overt complementizer under C. The reformulation of the ECP in (Chomsky, 1986) requires a large number of new concepts which we will not trace or pursue here. We retrace our steps to the *that*-trace filter (150), and ignore the problems raised by the subtle grammaticality differences of the parasitic gap constructions (154).

2.11 Concluding Remarks

We have presented an overview of the Government-binding theory in which serious attempt was made to try to capture, within the framework of attributed definitions, the mechanisms and principles of grammar proposed in the literature. A more detailed attribute grammar specification is given in Chapter 3. In sections 2.1 and 2.2 we showed that, under Government-binding assumptions, all vocabulary symbols of the grammar should be viewed as attributed/complex symbols, and that the dichotomy introduced in (Chomsky, 1965) between lexical categories and formatives on one hand, and other symbols on the other need not be maintained. We also hinted at the possibility of eliminating rule move-α from the grammar, reducing it to an interpretive Chain rule which determines trace-antecedent relations at S-Structure. This is accomplished in Chapter 3.

There are two major omissions in the discussion above: Control theory, and more detailed discussion of the rule Quantifier Raising and its role in the derivation of logical forms. This choice was made only to limit somewhat the scope of the discussion. It seems clear that exactly the same set of devices employed in this chapter to formulate symbolically the other components of the theory may be utilized to implement the two components omitted. Our principal interest in this thesis is the derivation of S-Structure and the formulation of an analysis procedure for the grammar. Omission of Control and quantifier raising do not hamper to any mayor extent the proposed goals, since both components apply only in the derivation of logical forms. The extent to which they affect the grammar obtained is the slight overgeneration their omission introduces.

We have for the most part ignored issues of Universal Grammar. It is our contention that the proposed framework of attributed definitions, with the particular content given here, and with some parametric factors added, is sufficient to state a grammar with the status of explanatory adequacy -- i.e., an Universal Grammar. We believe, however, that formulation of significant claims about Universal grammar (within the attributed framework adopted here) need wait at least until significant fragments of particular grammars for other languages appear (within the

same framework). In spite of the impressive depth and breath of explanation and description that Government-binding grammar achieves, there are still many unresolved issues and details of realization in the grammar of the particular language we consider, English. The further refinement of the grammar that we present in Chapter 3 should be viewed in this context.

Chapter 3. An Attribute-Grammar Specification of Government-binding Theory

The objective in the present chapter is to explore the use of attribute grammar as a framework for the definition of the grammar of natural languages, assuming the theoretical content of Government-binding. We are concerned, in particular, with the formulation of an attribute grammar for English, following the notions and principles of Government-binding. The scope of the discussion in the present chapter is narrower than that in Chapter 2, where a more general view of the theory is presented.

We do not attempt to formulate the grammar in a parametric fashion, approximating a characterization of what might be Universal Grammar (UG). This is for several reasons. The immediate data (or particular grammars) on which inductive hypotheses may be made about properties of UG and its parameters is scarce. Given that our framework of formalization is attribute grammar, it would be of most help to have available within the same framework the grammars of several natural languages, especially if the sample languages are representative of the typological differences in natural language. Although there are several parameters of UG that have been identified and have a fairly good chance of validity, such as the head-complement order in the X-Bar convention, the choice of bounding nodes that a language makes, and the "pro-drop" parameter, their number is small and the manner in which they may be incorporated into the grammar presented here seems obvious; it does not seem worth to complicate further the exposition and task at hand for the sake of these parameters.

An attribute grammar has a simple and direct interpretation as the specification of a non-deterministic automaton for the analysis and generation of strings in the language it defines.[1]

---

[1] For restricted classes of attribute grammars, the automaton can be guaranteed to be deterministic and derived mechanically from the attribute grammar (Kastens, Hutt, and Zimmermann, 1980). If we restrict the

In view of this property, attribute grammars developed along the lines suggested here are partial specifications of automata embodying the Government-binding theory of grammar. Attribute grammar specifies in detail the computations that these automata carry out in their course of operation analyzing or generating natural language strings. The present research is thus a move in the direction of seeking specific mechanisms and realizations of Universal Grammar. Because of this, we expect decisions and details of realization which are not dictated by any particular linguistic or psychological facts, but perhaps only by matters of style, and possible computational efficiency considerations for the final grammar. It is therefore safe to assume that the attribute grammar we obtain admits of several non-isomorphic variants, none of which is to be prefered over the others a priori. Consideration of similar specifications of parametrized grammars for different languages might provide the criteria on which alternatives may be evaluated, and may eventually lead to substantive generalizations about the computational mechanisms employed in natural languages.

Unlike earlier computational models of Government-binding (Berwick and Weinberg, 1984), whose statement is tightly bound to a particular parsing automaton (Marcus, 1980), the attribute grammar developed here is more abstract and neutral regarding the choice of parsing automata. To the extent that our implementation is correct, principles of Universal Grammar may be shown to follow from the automaton partially defined by the attribute grammar. In this light, principles of Universal Grammar are specifications about the operation of the automaton defined. Our view is that principles of grammar stand with respect to the attribute grammar given here in the same way that a high-level specification stands to a program that meets it.

This chapter is organized into six sections, as follows. In section 3.1 we define attribute grammars and prove that their generative power is Type-0. In section 3.2 we formulate a Chain rule, which replaces move-α in the standard Government-binding model. In section 3.3 we present the attribution that defines the processes of government, Case- and θ-marking. In section 3.4 we give two attribution rules that functionally classify empty nominals by their local context. In section 3.5 we review the facts about the English auxiliary and propose a mechanism to account for it without the standard affix-hopping transformation. In section 3.6 we provide a computational model of Binding, in which the generate-and-test approach of the free indexing

---

interpretation of attribute grammar rules in the manner of the augmented phrase structure grammars of Heidorn (1975), we may obtain efficient analysis and generation routines (Heidorn, 1972).

rule is refined into a more directed Binding rule. We close the chapter with some remarks on the grammar developed.

3.1 Attribute Grammars

Attribute grammars are an extension of context-free grammars, originally developed for the formal specification of the semantics of programming languages and compiler construction for the same. 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). In this section we define attribute grammars and introduce the main concepts and properties associated with them. The characterization of attribute grammar in the present section is adapted from Waite and Goos (1984).

3.1.1 Definition

An *attribute grammar* is defined using a context-free grammar $G=(N, T, P, Z)$, where N and T are non-terminal 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 or domain $dom(a)$ of possible values, for each attribute $a \in A(X)$.[2] Each attribute represents a specific, possibly context-sensitive property of symbol X. The value of each attribute occurrence $X.a$ in a derivation tree must be uniquely defined. Each attribute occurrence is a single-assignment variable.

Attribute values are defined by *attribution rules* of the form $X_i.a \leftarrow f(X_j.b, ..., X_k.c)$, 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 applicative expression whose value depends only of the values of attribute occurrences associated with other symbols in p. In particular, it is not assumed that f performs any kind of tree traversals to compute its value.[3] When p applies in a derivation, the attribution rule defines

---

[2] 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*).

[3] This intended characteristic of attribute grammars cannot be enforced in a vacuum, ignoring the *types* of attributes

the value of attribute occurrence $X_i.a$ in terms of the occurrences $X_j.b$, ..., X subk..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 $b(X_i.a, ..., X_j.b)$, $0 \leq i, j, \leq n$, on the attributes of symbols occurring in the production p. These conditions are truth-valued functions which must be satisfied in derivation trees requiring application of the production. The conditions contribute to the notion of *grammaticality* in the language generated by the attributed grammar. Structurally plausible constructions may be ruled out by 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 (1).

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

(i) G=(N, T, P, Z) is a context-free grammar,

(ii) $A=\bigcup_{X\in V} A(X)$ is a finite set of *attributes*,

(iii) $R=\bigcup_{p\in P} R(p)$ is a finite set of *attribution rules* of the form $X.a \leftarrow f(Y.b, ..., Z.c)$, and

(iv) $B=\bigcup_{p\in P} B(p)$ is a finite set of *attribute conditions* of the form $b(X.a, ..., Y.b)$.

The underlying context-free grammar G assigns a derivation tree to each sentence in L(G). Each node labelled X in the tree is annotated with the set $A(X)$ of attributes associated with X. Each attribute of $A(X)$ defines an *attribute occurrence* at node 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

---

allowed. For if pointer types are permited, 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.

yield *true*. We define the *language* generated by the attribute grammar, L(AG), as the set of sentences in L(G) which have at least one correctly attributed tree.[4]

For each production p in G, we let the set $AF(p) = \{ X_i.a : X_i.a \leftarrow f(...) \in R(p) \}$ denote 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 daugthers 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 below 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 (2).

(2) Let AG = (G, A, R, B) be an attribute grammar, X a grammar symbol, and $a \in A(X)$. The attribute occurrence X.a is *synthesized* if there exists a production $p = X \rightarrow \gamma$ such that X.$a \in AF(p)$. The attribute occurrence 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 attribute occurrences of X, as in (3).

(3) $AS(X) = \{ X.a : \text{ for some } p = X \rightarrow \gamma \in P, X.a \in AF(p) \}$

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

We admit other attributes, such as the features of a lexical item, whose values are determined not by rule, but by the symbol with which they are associated. We call these attributes *inherent* or intrinsic. This class may be reduced to the former two classes of attributes without loss of generality.

---

[4] We do not impose any condition on the degree of ambiguity of the underlying grammar G.

3.1.2 Well-formedness

The preceeding characterization of attribute grammar permits writing grammars with certain undesirable properties. Nothing thus far prevents grammars which yield improperly attributed trees. This situation occurs when some attribute occurrence in a derivation tree has its value defined multiple times (in possibly multiple ways), or does not have a value defined at all.

(4) 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 an 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; $AS(X) \cap AI(X) = \phi$. Further, 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 (Waite and Goos, 1984). Also, for the axiom Z of the grammar AI(Z) is empty and for each terminal symbol *t* (or formative) AS(*t*) is empty.

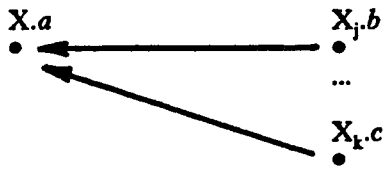
Well-formedness of an attribute grammar also depends on the nature of the data dependencies between attributes. Let X.*a* be an attribute occurrence whose value in a derivation tree is defined by the attribution rule $X.a \leftarrow f(X_j.b, ..., X_k.c)$. In order to compute the value of X.*a*, we must know the values of all attribute occurrences $X_j.b, ..., X_k.c$ upon which X.*a* depends.[5] In this case we say that X.*a directly depends* on attributes occurrences $X_j.b, ..., X_k.c$. This situation is shown graphically in the directed graph (5), where the node set is the set of attribute occurrences involved, and the edge set is the set of direct attribute dependencies. We let $D_{X.a}(p)$ denote the

---

[5] If attribute evaluation uses *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. See also Chapter 4.

set of attribute occurrences of symbols in p on which X.a depends. D(p) denotes the set of all attribute dependencies induced by the attribution of production p.

(5)



If τ is a derivation tree, the collection D(τ) of attribute dependencies in τ is obtained by union of the collections D(p) of dependency relations induced by the productions used to derive the tree. Given a tree τ with node set K and collection D of direct dependency relations between attribute occurrences, a *dependency graph* $DT(\tau) = \langle \bigcup_{X \in K} A(X), D \rangle$ is defined which records all the dependency relations that arise form the attribution rules associated with productions applied to derive the tree. The node set of the dependency graph is the set of attribute occurrences in the tree; the edge set is determined by the dependency relations between the occurrences. We are now in a position to define a *well-defined* attribute grammar, as in (6).

(6) An attribute grammar is *well-defined* if and only if it is consistent, complete, and the dependency graph DT(τ) is acyclic for each derivation tree τ corresponding to a sentence of L(G).

Since attribute grammars allow the use of both inherited and synthesized attributes, it is not always obvious when the semantic rules do not amount to a circular definition (Knuth, 1968). An attribute grammar is said to be *circular* if DT(τ) has a cycle, for some derivation tree τ of G. Circularity is a decidable property of attribute grammar, as shown by (Knuth, 1968). However, the complexity of the test is inherently exponential in grammar size (Jazayeri, Odgen, and Rounds, 1975; Jazayeri, 1981). Examples of the dependency graphs that arise from our attribute grammar appear in Chapter 4. The definition (6) assumes *value semantics* for attribute evaluation.

3.1.3 Attribution algorithms

If an attribute grammar is well-defined, a non-deterministic algorithm can be used to compute the attribute values in each derivation tree generated by G: For each occurrence X.*a* of an attribute in the derivation tree we provide a separate process to compute its value; the process is started when the values of all attribute occurrences on which X.*a* depends are known. When the current process completes, the value of X.*a* becomes available to other processes which depend on this value. The number of processes that is attached to a given tree depends on the number of attribute occurrences in the tree.

While the above scheme for attribute evaluation is conceptually simple and has a straightforward implementation in a language for concurrent computation, it may not be useful in practice due to its non-deterministic nature and the requirement that a derivation tree be available before attribute evaluation begins. Due to the typical extreme ambiguity of the underlying syntax of natural language grammars, it is highly desirable to combine attribute evaluation and condition testing with derivation actions. This technique is known as *attribute-directed* parsing (Watt, 1980). It permits early detection of derivations that ultimately lead to incorrectly attributed trees.[6] Typical attribution schemes for programming languages assume restricted classes of attribute grammar. On goal of these restrictions is to permit attribute evaluation by a simple sequence of tree-traversals, which may be determined statically, independent of any input, at evaluator construction time. For each production $p = X_0 \rightarrow X_1 \ldots X_n$ a fixed *attribution algorithm* may be defined, consisting of a sequence of the basic tree-walk steps in (7).

(7) a. Evaluate a rule or condition from $R(p) \cup B(p)$

b. Move to child node X_i, $i = 1, \ldots, n$

c. Return to the parent node X_0

---

[6] Ambiguity in natural language grammar is the more challenging technical problem. Storage requirements are far less severe than in programming languages, since typical sentence (program) input in the latter may be in the order of tens of thousands of tokens, as opposed to the far shorter 20 to 30 words in typical natural language utterances. It is possible to store an entire attributed tree for a natural language utterance in the available working memory of the automaton.

We postpone further discussion of attribute evaluation until Chapter 4.

3.1.4 Generative power of attribute grammars

Kimball (1967) and Peters and Ritchie (1973) showed that the generative power of the transformational grammars admitted in the Standard Theory is equal to that of Type-0 grammars, even if the base is limited to context-free or regular form. A similar result may be provable about Government-binding grammars.[7] We may thus question whether the class of languages that may be described by GB grammars may also be described by means of attribute grammars -- i.e., whether attribute grammars are sufficiently powerful tools for the description of natural language.

The generative power of attribute grammars has not been investigated in much detail, perhaps due to the elaborate grammatical model they define, and the fact that the semantic domains of attributes and the kind of attribution functions and conditions allowed have been left largely unspecified (Engelfriet and Filè, 1981). In the original definition of attribute grammar (Knuth, 1968), and also in more recent ones (Waite and Goos, 1984), it is not indicated how to define the domains of attributes or the functions that may appear in the attribution. In Knuth (1968), "the semantic rules are expressed in terms of notations which are assumed to be already understood" -- i.e., with independent semantics; Waite and Goos (1981), on the other hand, simply require that "all attributes [be] effectively computable."

It is clear that the generative power of attribute grammar is at least that of the kind of attribution functions allowed in the grammars. For example, if partial recursive functions are permitted as attribution functions, then AG has trivially the formal power of Type-0 grammars. Theorem 1 shows that attribute grammars have Type-0 generative power, even if the attribute domains and functions are severely restricted. Theorem 1 answers our original question in the affirmative, in a less trivial sense, provided natural languages are recursively enumerable.

---

[7] We are not interested in this problem, since its answer is likely to depend to a large extent on the particular formalization chosen of the already complex model of Government-binding grammar, and it is not clear that establishing the result will further our search for particular analysis procedures (although it would mean that no general analysis procedure exists).

Theorem 1: Attribute grammars generate the Type-0 languages.

Proof: [8] The proof is by reduction of an arbitrary single-tape (deterministic) Turing machine $M=(Q, \Sigma, \Gamma, \delta, q_0, \mathfrak{b}, F)$ to an attribute grammar $AG=(G, A, R, B)$, such that the language accepted by M is equal to the language defined by AG. Q is a finite set of states, Γ a finite vocabulary of tape symbols, $\Sigma \subset \Gamma$ a finite vocabulary of input symbols, δ: $Q\times\Gamma\rightarrow Q\times\Gamma\times\{L, R\}$ the next move mapping, $q_0 \in Q$ the initial state, $\mathfrak{b} \in \Gamma-\Sigma$ the blank symbol, and $F \subseteq Q$ a set of final states. The machine has a single tape infinite in both directions, and its head is initially positioned at the zero-th cell of the tape.

The underlying grammar is $G=(N, T, P, Z)$, with $N=\{A, Z\} \cup \{q_a : q \in Q \text{ and } a \in \Gamma\}$, $T=\Sigma$, and the production set P given by (8).

(8) $P=P_0 \cup P_1 \cup P_2$, where

$$P_0=\{Z \rightarrow A\ q_{0a} :\ a \in \Gamma\}$$

$$P_1=\{A \rightarrow \epsilon\} \cup \{A \rightarrow t\ A :\ t \in \Sigma\}$$

$$P_2=\{q_a \rightarrow q_x' :\ x \in \Gamma \text{ and } qa\rightarrow q'bD \in \delta, \text{ for some } b \in \Gamma \text{ and } D=L,R\}$$
$$\cup \{f_a \rightarrow \epsilon :\ a \in \Gamma \text{ and } f \in F\}$$

Assume a synthesized attribute s associated with A, and inherited attributes l and r associated with non-terminals of the form q_a. The domain of each attribute is Γ^*. The value of an attribute occurrence $A.s$ at a node x in a derivation tree is the terminal string dominated by x. M is simulated by attributed derivations with non-terminals q_a, employing unit and ϵ-productions in P_2. An *instantaneous description* of M is encoded in an attributed symbol q_a with associated attribute occurrences $q_a.l$ and $q_a.r$: we let q denote the current state, $q_a.l$ the tape contents to the left of the head, in reverse order, and $q_a.r$ the tape contents to the right of the head, including the head position. The attribution appears in schema form in (9).

---

[8] I thank Francisco Corella for pointing out an error in an earlier version of the proof.

(9) a. $Z \rightarrow A\ q_{0a}$, for $a \in \Gamma$

attribution:

$q_{0a}.l \leftarrow \epsilon$

$q_{0a}.r \leftarrow A.s$

b. $A \rightarrow \epsilon$

attribution:

$A.s \leftarrow \epsilon$

c. $A \rightarrow t\ A$, for $t \in \Sigma$

attribution:

$A_1.s \leftarrow t \parallel A_2.s$

d. $q_a \rightarrow q_x'$, for $q, q' \in Q$, $a, x \in \Gamma$

attribution:

(Move left: $qa \rightarrow q'bL \in \delta$)

$q_x'.l \leftarrow tail(q_a.l)$

$q_x'.r \leftarrow front(q_a.l) \parallel b \parallel tail(q_a.r)$

(Move right: $qa \rightarrow q'bR \in \delta$)

$q_x'.l \leftarrow b \parallel q_a.l$

$q_x'.r \leftarrow tail(q_a.r)$

condition:

$front(q_a.r) = a$

If $a \in \Gamma$ and $\omega \in \Gamma^*$, the expression $a \parallel \omega$ denotes the string obtained from ω by placing symbol a in front of it; $tail(\omega)$ denotes the string obtained from ω by removing its first symbol, if $\omega \neq \epsilon$; $front(\omega)$ denotes the first symbol of ω, if $\omega \neq \epsilon$. If $\omega = \epsilon$, then $tail(\omega) = \epsilon$ and $front(\omega) = b$, the blank symbol in Γ. Since M is deterministic, by hypothesis, each pair $<q,a> \in Q \times \Gamma$ determines at most one element in δ and only one of the alternatives in the attribution (2.d) applies.

By construction of AG, a non-terminal q_a may derive the empty string after $n \geq 1$ steps if and only if there is some sequence of states $q_i \in Q$ and symbols $a_i \in \Gamma$, $i = 1, ..., n$, such that $q = q_1$, $a = a_1$, $q_n \in F$, and $q_j a_j \rightarrow q_{j+1} x D \in \delta$, for $x \in \Gamma$, $D = L$ or R, and $j = 1, ..., n\text{-}1$. However, assuming values α and β for the attribute occurrences $q_a.l$ and $q_a.r$ associated with q_a, the derivation from

q_a defines a *correctly attributed* tree if and only if the sequence of moves defined by the q_i and a_i in the derivation causes M to enter a final state, when starting in ID $\alpha q \beta$. In particular, considering a string $\omega \in \Sigma^*$ generated by G and non-terminals nodes A and q_{0a} introduced by application of a production in P_0, where A dominates ω, we have $q_{0a}.l = \epsilon$ and $q_{0a}.r = \omega$. Hence, ω has a correctly attributed tree if and only if $\omega \in L(M)$. We conclude $L(AG) = L(M)$. □[9]

Theorem 1 shows that the method of attribute definition is as powerful as any method for enumeration of recursively enumerable sets, even if the semantic domains are limited to a simple string domain and the attribution functions to basic operations on that domain -- that is, the primitive operations needed to manipulate the IDs of a Turing machine. The Type-0 generative power of attribute grammars should not be seen as a drawback as tools for linguistic analysis. They have significantly more structure that unrestricted rewriting systems and provide a convenient structural description for each string in the languages they generate. The reduction of the machine to an attribute grammar in the theorem is reminiscent of the reductions by (Koster, 1971) and (Deransart and Maluszynski, 1985) of a Turing machine to an affix grammar and a definite clause program, respectively.

An interesting result about attribute grammar follows: There is no general decision procedure to test membership of a string ω in the language generated by an attribute grammar AG, for arbitrary grammar AG and string ω. The reason has to do with the potential degree of ambiguity of the context-free grammar underlying AG. For the degree of ambiguity may be infinite, and the decision procedure may have to enumerate all possible analyses to evaluate the attribution. This is the case in the construction of Theorem 1, where derivations from the non-terminals q_a may be non-terminating. If the underlying grammar is finitely ambiguous and the attribution is well-defined, then the membership problem for the resulting class of grammars is decidable. For each string, we enumerate its (finitely many) structural descriptions, apply the attribution rules and conditions, and accept if at least one tree is correctly attributed.

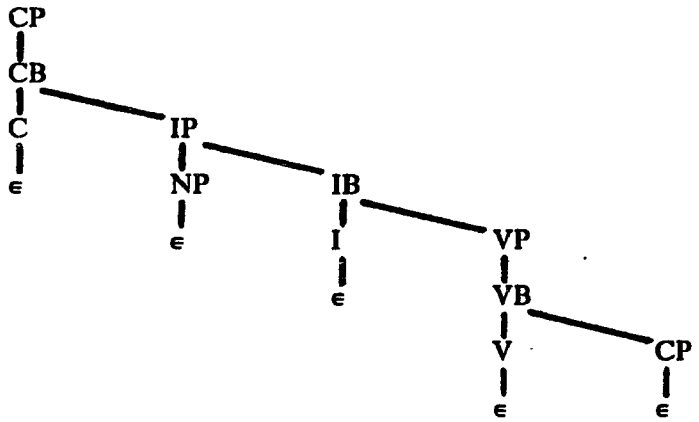
It is of interest to note that the phrase structure component of current Government-binding grammar is infinitely ambiguous, chiefly due to the extremely prolific use made of ϵ-productions -- or, correspondingly, transformations that apply freely to introduce empty categories. We illustrate this with the symbol CP, which is recursive on itself, as shown in the typical derivation

---

[9] We note that the grammar G of the construction in the theorem need not be reduced.

tree (10). Recursion in (10) is via the position allowed for clausal complements of a verb. Each of the leaves C, NP, I, V, and CP in the tree is possibly empty, as we saw in Chapter 2. Hence, it is possible to derive infinitely many CP structures with (10) as the basic recursive unit. Each structure yields the empty string.

(10)



The ambiguity of the phrase structure component, however, is constrained by the subtheories in Government-binding. The relevant constraints may be captured by attribution rules and conditions that rule out any CP structure which yields the empty string, unless it is the simple structure $[_{CP}\ \epsilon]$.[10] The base grammar we define in Chapter 4 solves the problem noted with the structure (10) by requiring an overt verbal element in every clause.

3.2 Accounting for Movement in the Grammar

We now proceed to give an attribute grammar formulation of Government-binding theory, beginning with an interpretive account of movement.

3.2.1 Interpretive models

We assume a variant of Government-binding in which there is no movement transformation or,

---

10 This is only a substantive property of Government-binding grammar, for which I have no formal proof.

for that matter, any other syntactic transformation that applies in the derivation of S-Structure.[11] We develop a grammatical model in which the operations formerly carried out by transformations reduce to attribute computations on phrase-structure trees.

3.2.1.1 Levels of representation in an interpretive theory

In the grammatical model we assume, context-free rules generate trees at a level of representation which, reverting to older terminology, we call *surface structure* (ss). This level is identical to S-Structure, except for the fact that the association between dislocated phrases and their traces is not present.[12] The base generates overt phrases in their surface places, along with empty categories where appropriate. Chain indices that reveal history of movement in the transformational account are, however, not present. Government relations are identified at surface structure. Similarly, Case-marking and thematic role assignment take place at surface structure. But thematic roles and Case may end assigned to an empty category which is not yet identified with its antecedent, if it has one. Functional classification of empty categories is possible at surface structure, as discussed in section 2.6.4, without reference to their antecedents.

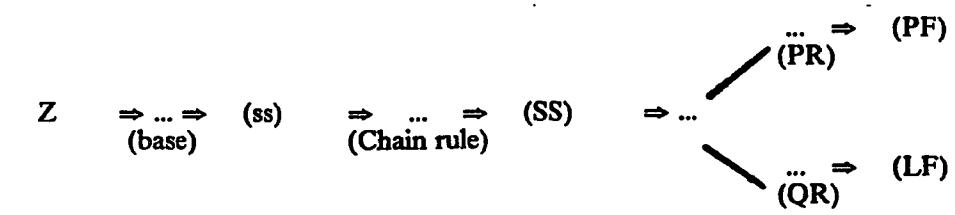
We define an interpretive *Chain rule* which defines syntactic chains interpretively, by linking traces with their antecedents, yielding the S-Structure level. The interpretive notion of S-Structure that we obtain is identical to the transformational one, as in Chapter 2 (assuming that the Chain rule and move-α may be proved to be empirically equivalent). Phonetic Form and Logical From are derived from S-Structure, possibly as before, and we obtain the organization (11) of the grammar into levels of representation.

---

[11] We leave open the question of what happens in the derivation of PF and LF. The derivation of LF depends on the treatment of quantification adopted. In section 3.4 we show that "affix-hopping" may be reduced to attribute computation. By the remarks in Chapter 2, other PF transformations like NP-shift may too be eliminated. Topicalization and clefting are handled by a slight extension of the Chain rule in section 3.2.7.

[12] We choose the name *surface structure*, rather than keep the D-Structure, since representations at surface structure correspond closely to the surface form of utterances.

(11)



Most principles of grammar apply as before; however, the Projection principle of thematic theory does not apply at surface structure, but only at S-Structure and LF.

3.2.1.2 Previous interpretive models

The elimination of transformations from the grammar is not a new idea; it has in fact been proposed at various times. Koster (1978) was perhaps the first to propose the transformation-less alternative,[13] arguing that the distinction assumed between *trace* and base-generated PRO in transformational grammar is not necessary. This fact would make it possible to apply the rules of interpretation used to identify PRO antecedents to establish trace antecedents. Consideration of Koster's proposal in Government-binding is anachronistic, given the well-established distinction between trace and PRO in the latter. Also, even if we ignore the distinction, Control theory is not developed to the same level of detail that the theory of movement is.

Chomsky (1981, p.89-92) refers to an alternative theory in which S-Structure is generated directly by the base. Rule move-α is eliminated in favor of interpretive rules in the Logical Form component. After these rules apply, they establish trace-antecedent relations. The output of the rules is subject to the conditions (12), and perhaps some others not discussed by Chomsky. We may assume a generate-and-test approach for the operation of the LF rules, similar to the free-indexing rule of the Binding theory. Conditions on empty categories are stipulated as filters on the output of the rule (cf. section 2.6.4).

(12) If (α_i, β_i) are coindexed, where α is antecedent and β trace, then

(i) α_i lacks and independent θ-role -- i.e., it is in a θ-Bar position.

---

[13] I thank Bob Berwick for pointing this out to me.

(ii) β_i is properly governed

(iii) the relation is subject to the Bounding theory -- i.e., α_i and β_i are subjacent.

Chomsky's proposal is close to the model (11) above, except for a renaming of levels of representation and assignment of the trace indexing rules to LF. The alternative (11) maintains the levels of S-Structure, LF, and PF as in the transformational theory. The *Chain* rule we define incorporates well-formedness conditions of chains in its operation, and does not employ the generate-and-test method implicit in Chomsky's proposal. Properties of chains, such as Subjacency and the Path Containment condition, are shown to follow from the definition of the Chain rule.

The most recent proposal we are aware of within the framework of GB linguistics to reduce the role of transformations in the grammar is due to Barss (1983).[14] Barss, like (Koster, 1978), adopts an interpretive (or representational) view of chains, ignoring the move-α mapping from D- to S-Structure. Chains are constructed by an interpretive algorithm at S-Structure.[15] Barss's algorithm is generate-and-test, like Chomsky's, but unlike the latter, it applies at S-Structure and has gives in detail the well-formedness conditions on non-argument chains. See Barss (1983) for further details.

In Generalized Phrase-Structure Grammar (GPSG), Gazdar et al. (1985) suggest that natural language phenomena might be accounted for by purely context-free means. GPSG has weak context-free generative power; however, the implication that GPSG uses strictly context-free means and can be parsed efficiently is not valid. A central component of GPSG is a system of syntactic features which, as we proved in section 3.1.4, extends easily the generative power of the grammar to that of Type-0 systems.[16] Ristad (1986) proves that GPSG recognition is exponential-polynomial hard in grammar size, which implies that the class of devices assumed in GPSG cannot be reduced to context-free ones without extraordinary (exponential) increase in

---

[14] I thank Prof. Kuno for pointing out this manuscript to me.

[15] Barss assumes move-α without the trace-convention for the derivation of S-Structure.

[16] We do not claim that GPSG grammar has Type-0 generative power.

the size of the resulting CFG. Gap-antecedent relations are established in GPSG by the category-valued feature SLASH. This feature is associated with nodes on the path between the antecedent and the gap, in a manner similar to the attribution done by the Chain rule below. However, unlike the use of the SLASH feature in GPSG to define long-distance dependencies, the Chain rule we present follows the theoretical assumptions of GB. It distinguishes two types of chain, argument and non-argument, and implements the well-formedness constraints that the GB subtheories define on chains.

In one of the early attempts to provide a recognition procedure for transformational grammar, Petrick (1965) devised an algorithm which operates as follows. Given a transformational grammar consisting of a (CFG) base and a finite set of arbitrary transformations, the algorithm first computes a context-free grammar capable of deriving all surface structures that may be generated by the original transformational grammar. The algorithm also computes inverses of the original transformations; these are maps from surface to deep structure. Given an input string, the runtime part of the algorithm uses the extended CFG to hypothesize surface structures, which are then checked for validity by the transformational component. By the trace theory in GB grammar, the information present at D-Structure may be inferred from S-Structure. In particular, it is possible to test whether a given S-Structure has a corresponding valid D-Structure. Thus, some updating and application of Petrick's procedures can provide an alternate characterization of Government-binding in which surface structure is base-generated and annotations and conditions on it are computed interpretively, like in all the previous proposals.

3.2.2 Trace indices

We consider now the trace indices employed below in the Chain rule and our conventions about them. We discuss the distinction we make between trace and referential indices. Associated with every node in a surface structure tree is an attribute *node* whose value is a unique positive integer and gives the preorder position of the node in the tree. In this manner we can refer unambiguously to any tree node. For each category that may be moved we also define an attribute *Chain*, whose value is a positive integer and identifies the syntactic chain to which the subtree dominated by that category belongs. We use positive integers to identify chains; the use of the *Chain* attribute is identical to that of the trace-convention in Chapter 2. Extending the terminology slightly, we also speak of the *node* or *Chain* number of a phrase and identify it with the *node* or *Chain* number of the root of the phrase.

Somewhat arbitrarily and for the sake of concreteness we assume that the *Chain* number of a moved phrase and all its traces is equal to the *node* number of the moved phrase in surface position. The *Chain* number of an unmoved phrase is the *node* number of that phrase. In what follows in this section we make some simplifications and limit the scope of movement to NP phrases, covering so-called NP- and Wh-movement. Thus, the attribute *Chain* is defined only for NP. But the manner in which chains (movement) may be determined interpretively for other categories should be clear; this involves a simple generalization of the Chain rule.

The *Chain* index introduced above is distinct from the referential index *RefIndex* of noun phrases. The latter defines anaphor- and pronominal-antecedent relations, according to the principles of control and binding theories. There is in principle no notational problem for representing the indices of corefering phrases in different chains, as in (13). The first element in each index pair is the *Chain* index of the phrase, while the second is the referential index. Thus ($John_{i,k}$) and ($he_{j,k}$, $[\in]_{j,k}$) are different chains, while the referential index k indicates that the three noun phrases corefer. We do not consider here the Logical Form mechanisms that may determine which phrase among coreferential ones determines the reading of the collection. Note that these mechanisms are not yet specified by any of the current GB subtheories (Higginbotham).

(13) $John_{i,k}$ said $[_{CP}$ that $he_{j,k}$ was struck $[\in]_{j,k}$ by lighting]

Introduction of a referential index distinct from the *Chain* index already assumed, or viceversa, yields redundancy and perhaps excess mechanism in the movement, Binding, and Control theories. It is often noted that trace-antecedent relations are subject to the same binding conditions as over anaphors. However, Binding theory is about argument-binding. It does not apply to Wh-traces. An extension of Binding to non-argument binding is required to unify "movement" and bound anaphora relations (Aoun, 1985). Pending further discussion, we assume that the value of the referential index on a number of correfering NPs is the *node* number of the lowest (*node*) numbered NP in the collection. If an NP has no antecedent, we set its *RefIndex* to its own *node* number; this condition represents the fact that the NP is not coindexed with any other NP.

To illustrate the use of *node*, *Chain*, and referential indices, let us consider S-Structure (14), assuming for simplicity that $NP_i.node = i$, for $i = 1, 2, 3$. Since NP_1 is at its D-Structure position, we have $NP_1.Chain = 1$. On the other hand, NP_2 heads the chain (NP_2, NP_3), which by our

Chain convention is identified as chain 2. Thus, $NP_2.Chain = NP_3.Chain = 2$. The referential indices are assigned as follows. Since NP_1 (*Mary*) is the the first noun phrase, $NP_1.RefIndex = 1$. Now, if we construe the pronoun *she* coreferential with *Mary*, we get $NP_2.RefIndex = NP_3.RefIndex = 1$; otherwise this value is set to 2.

(14) $[_{NP_1}$ Mary] said that $[_{NP_2}$ she] seems $[_{NP_3} \in]$ to be ready to take the job.

3.2.3 Properties of chains

Once surface structure is generated by the base, the interpretive Chain rule evaluates the *Chain* indices of noun phrases, thus deriving S-Structure. Properties of chains, which follow from the formulation of move-α, constraints on movement (e.g., subjacency), the ECP, Case, theta, and binding theories, may be used advantageously in the study and formulation of the Chain rule. In fact, once the interpretive rule is formulated, given that it represents computational mechanisms used in sentence understanding and to which certain psychological import may be attached, it may be more appropriate to consider the noted properties of chains to be abstract consequences of the interpretive rule, and not first principles of linguistic competence. If the interpretive rule, perhaps parametrized in some ways, turns out to be universal, we will have achieved (part of) an explanatorily adequate computational specification of linguistic competence, which it seems is to be preferred over an axiomatic one, on grounds of psychological reality. This is discussed further in Chapter 5.

In what follows, we limit our attention to chains that may be formed by movement up to the level of S-Structure. LF movement is easily accomplished by Quantifier-raising, taken as a tree-transducer between S-Structure and LF. Alternatively, we may develop an interpretive account of quantifier scope, which would replace Quantifier raising. We start with the classification (15) of chains.

(15) a. *Elementary chain* (contains a single element)

b. *Argument chain* (A-chain)

c. *Non-argument chain* (A-Bar chain)

An elementary chain is a chain whose single NP element has identical D- and S-Structure positions. This is a special instance of argument chain. If the NP is lexical (i.e., non-empty), this position must be Case-marked, and if the element is not expletive it must be also θ-marked. This follows from the Case filter and the Theta-criterion, respectively. If the NP is empty, it cannot be a trace (variable or NP-trace), and hence it must be PRO, in which case the position it occupies is ungoverned and θ-marked.[17] These two cases exhaust the possibilities for elementary chains.

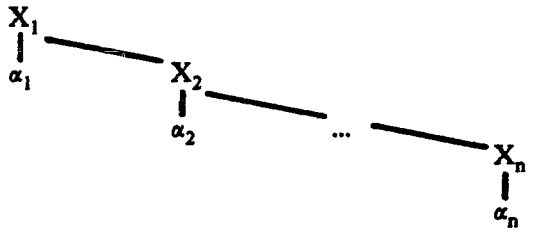
Argument chains are characterized by their last element, which is a trace in a θ-marked, but Case-less position. The θ-criterion and Projection Principle determine that every other element of the chain, including the head, must be in a non-θ marked argument position (θ-Bar position). The mechanism for thematic role assignment determines that Subject position (specifier of I) is the only position that may satisfy these two conditions. Case theory requires that exactly one position of the chain be Case-marked; Chomsky (1986a) assumes that this position is always the head.

A non-argument (or operator) chain is headed by an overt Wh-phrase or an empty operator in C-specifier position. The interpretation of this phrase or operator is defined by the LF component. Every other element of the chain, except the last, is a trace, also in C-specifier position. The last element is a trace in Case-marked position, and is interpreted as a logical variable, bound by the operator that heads the chain.

An important property of movement that we simply assume is stipulated, not derived from more basic conditions, is that movement is always to a c-commanding position. The notion of c-command is defined in Chapter 2, in (72). By the condition, adjacent chain elements stand in a c-command relation: in chain $C=(\alpha_1, ..., \alpha_n)$, α_i c-commands α_{i+1}, for $i=1, ..., n\text{-}1$. The c-command condition implies that the chain C defines a binary-branching structure (16), which is the subgraph of the tree containing every element α_i of the chain. For each i, X_i is the first branching node dominating α_i. We refer to the path $X_1, ..., X_n$ in (16) as the *hub* of the chain. By the Subjacency condition (142.c) in Chapter 2, which constitutes Bounding theory as it applies to movement, adjacent chain positions are also subjacent. The path between adjacent hub elements, excluding the top-most node, may contain at most one bounding node.

---

[17] We ignore small "pro".

(16) 

A computationally important property of (16) is that X_i immediately dominates α_i, for $i = 1, ..., n$. This property of chains is important for the attribution of the Chain rule in the next section, since chains are propagated along hub nodes. The property permits identification of chain elements α_i locally in the tree, as immediate descendents of hub nodes. This eliminates the need for upward propagation in the tree of references to trace antecedents. The property does not follow from either the c-command or subjacency conditions, but can be readily verified by inspection on possible cases of movement. NP-movement is always to I specifier (or Subject) position, which is immediately dominated by branching IP node. Wh-movement is to C-specifier position, which is immediately dominated by CP. Similarly, the extraction site α_n is either Subject (of I) or complement (of V and possibly P) position. The productions responsible for generating the respective landing and extraction positions are discussed in Chapter 2, section 2.3.

3.2.4 Interpretive Chain rule

Given the above remarks about properties of chains, the following construction can be shown to implement the interpretive rule needed for chain determination. First, assume a pair of attributes *A-Chain* and *AB-Chain* associated with every syntactic category that may label a node in the c-command domain of NP. In particular, these attributes are associated with CP, CB, IP, IB, VP, and VB. Also, assume that chains are identified by positive integers. From the principles of the transformational cycle, the doubly-filled COMP filter, and the Subjacency condition, at most one argument and one non-argument chain may propagate across any given node on a chain-hub.[18] Thus, the attributes *A-Chain* and *AB-Chain* may identify the A- and A-Bar chains that propagate across the given node, if any. An appropriate domain for the two attributes is the natural numbers. Our grammar assigns to occurrences of *A-Chain* and *AB-Chain* at a given node in a derivation tree the number of the argument or non-argument chain, respectively, that

---

[18] Multiple Wh-extraction is possible if the doubly-filled COMP filter or the Subjacency condition are relaxed. We return to this point in section 3.2.6.

propagates across that node, or zero, if there is no chain across the node. For the matrix CP node, the the *A-Chain* and *AB-Chain* attributes have the value zero.

The interpretive rule includes mechanisms to identify the first, intermediate, and last elements of chains. Let us first consider argument chains. Following Chomsky (1986a), what distinguishes an A-chain head is that it is in a θ-Bar position; it is either Case-marked or ungoverned. If the position is Case-marked, the NP occupying it may be lexical (17.a), or a Wh-trace (17.b), in which case it is also the last element of an A-Bar chain. If the position is ungoverned, the element must be PRO (17.c). Intermediate positions of the A-chain are Case-less and θ-Bar argument positions, while the last position is Case-less, but θ-marked.

(17) a. John$_i$ seemed [ϵ]$_i$ to be certain [ϵ]$_i$ to win.

b. Who$_i$ [$_{IP}$ [ϵ]$_i$ seemed [ϵ]$_i$ to be certain [ϵ]$_i$ to win]

c. Everybody likes [$_{IP}$ PRO$_i$ to be loved [ϵ]$_i$]

Non-argument chains are simpler; they are headed by a Wh-phrase or empty operator in C-specifier position.[19] Intermediate positions are all Wh-traces in C-specifier position, and the last element is a Wh-trace, in Case-marked position. If the last element is in a θ-position, as in (18.a), the A-Bar chain does not intersect with any other chain; if the position is θ-Bar (17.b), the last element is shared with an argument chain ending at that position. Example (18.b) illustrates the more complex case of two disjoint chains, but which overlap their hub nodes, involving A- and A-Bar movement.

(18) a. Who$_i$ did [$_{IP}$ John say [$_{CP}$ [ϵ]$_i$ that [$_{IP}$ Bill fired [ϵ]$_i$]]]

b. Who$_i$ did John$_j$ seem [$_{CP}$ [ϵ]$_i$ [$_{IP}$ [ϵ]$_j$ to love [ϵ]$_i$]]

A precise statement of the attribution rules which define the core cases of the Chain rule is given in (19) and (20.a-e) below. (19.a) is a general attribution rule that assigns NPs to a chain (sets the value of the *Chain* attribute of the NP), assuming the values of the attributes *A-Chain* and

---

[19] We assume empty operators are generated optionally in C-specifier position.

AB-Chain at the node are already defined. The rule relies heavily on having the functional type of the NP determined. (19.b) is an attribute condition on chains, which requires chain termination and binding of traces. The attribution sets NP.*Chain* to zero only if NP is a trace and has no potential antecedent (NP.*A-Chain* = NP.*AB-Chain* = 0).

(19) **Chain rule: assignment of NP to a chain**

a. attribution:

NP.*Chain* ← if NP.*empty* = '−' then NP.*node*
else if NP.*pronominal* = '+' then NP.*node*
else if NP.*anaphoric* = '+' then NP.*A-Chain*
else NP.*AB-Chain*

b. condition:

NP.*Chain* ≠ 0

The chain percolation rules (20) are more involved. They define the value of the *A-Chain* and *AB-Chain* attributes at all nodes in an S-Structure tree, starting at the root node. The chain information carried by the attributes is propagated downwards. A brief informal sketch is as follows: For the root node CP the attribution rules (20.a) dictate CP*A-Chain* = CP*AB-Chain* = 0. The values of the two attributes are then essentially percolated downwards. However, when a C-specifier position is found (20.b), a non-argument chain is started by setting the value of the CB.*AB-Chain* attribute to the NP.*Chain* number of the position. The non-zero value of the attribute identifies the new A-Bar chain and is propagated downwards. Wh-traces in the c-command domain of the position are then able to identify their antecedent, when the general attribution rule (19.a) applies. Similarly, when a θ-Bar position is found (20.c), filled by a non-expletive, an argument chain is started, setting the value of the *A-Chain* attribute to the *node* number of the position. Lower NP-traces may pick up their antecedent as before. A chain is terminated when its last element is found. The attribute condition in (20.b) requires intermediate traces of A-Bar chains to be present.

(20) **Chain rule: chain propagation**

a. Start production

Z → CP

attribution:

CP.*A-Chain* ← 0

CP.*AB-Chain* ← 0

b. Clause productions

CP → CB

attribution:

CB.x ← CP.x, for x = *A-Chain*, *AB-Chain*

condition:

CP.*AB-Chain* = 0

CP → NP CB

attribution:

NP.x ← CP.x, for x = *A-Chain*, *AB-Chain*

CB.*A-Chain* ← CP.*A-Chain*

CB.*AB-Chain* ← NP.*Chain*

CB → C IP

attribution:

IP.x ← CB.x, for x = *A-Chain*, *AB-Chain*

c. Sentence productions

IP → NP IB

attribution:

NP.x ← IP.x, for x = *A-Chain*, *AB-Chain*

IB.*A-Chain* ← if NP.θ = *nil* & NP.*expl* = '−' then NP.*Chain* else 0

IB.*AB-Chain* ← if NP.*empty* = '−' or NP.*Case* = *nil* then IP.*AB-Chain* else 0

IB → I VP

attribution:

VP.x ← IB.x, for x = *A-Chain*, *AB-Chain*

d. Verb-phrase productions

VP → ... VB ...

attribution:

VB.x ← VP.x, for x = *A-Chain*, *AB-Chain*

VB → V
condition:
VB.x = 0, for x = *A-Chain*, *AB-Chain*

VB → V XP
attribution:
XP.x ← VB.x, for x = *A-Chain*, *AB-Chain*

e. Noun-phrase productions
NP → NP_1 NB
attribution:
NB.x ← 0, for x = *A-Chain*, *AB-Chain*
NP_1.x ← 0, for x = *A-Chain*, *AB-Chain*

NB → N XP
attribution:
XP.x ← NB.x, for x = *A-Chain*, *AB-Chain*

Downward propagation of *A-Chain* and *AB-Chain* takes place along chain hubs and enforces in a simple way the c-command condition between a trace and its antecedent. Chain elements are detected from the hub node that immediately dominates the element. Attribution rules (20.a-e) can be defined locally on each production whose left-hand side is a potential hub element. In the formulation of the attribution rules, particularly (19), it is assumed that certain other components of Government-binding have already been implemented, in particular parts of Government and Case theories, which contribute to the functional determination of empty categories. The implementation of the relevant parts of these subtheories is described in section 3.3 below; the attributes *empty*, *pronominal*, and *anaphoric* are set by the attribution rules discussed there, and achieve a full partitioning of NP, according to functional type (van Riemsdijk and Williams, 1986).

The attributes *Case* and *Gov* are inherited attributes of NP. Their values may be determined independently of the attributes *Chain*, *pronominal*, and *anaphoric*. *Case* records the grammatical Case that is assigned to the NP by the lexical element or configuration that Case-marks it; the value of the attribute is *nil* if the NP receives no Case. On the other hand, the value of *Gov* is an ordered pair consisting of a natural number and one of the values *nil* or *proper*. If the NP has a

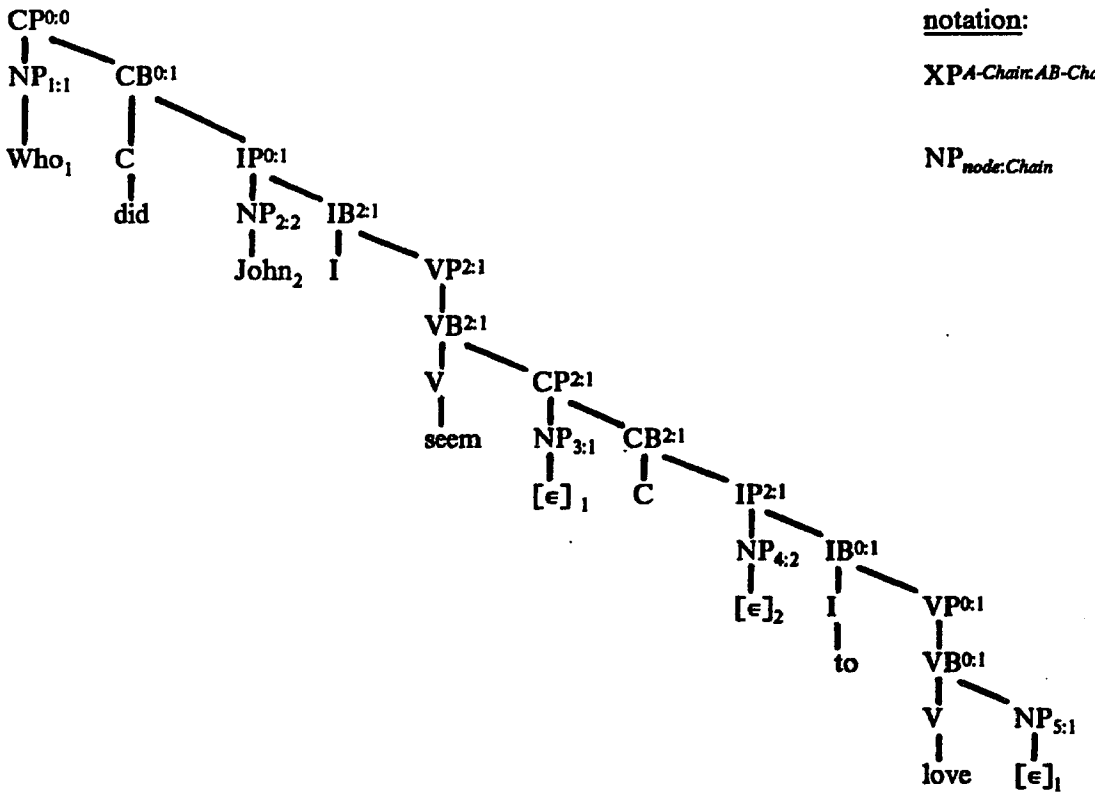
governor, the first element is the *node* number of the governor; otherwise the value is zero. The second element indicates whether the governor, if present, is proper or not. The attribution that defines the value of the *anaphoric* and *pronominal* attributes is simply a restatement of table (113) in Chapter 2.

The claim associated with the Chain rule defined by (19) and (20) is that given a *surface structure* in the sense of section 3.1, it derives the correct trace-antecedent relations after it applies.[20] An illustrative sample of its operation is shown in (21), where the structure tree of sentence (18.b) is shown. The annotations superscripted to the CP, CB, IP, IB, VP, and VB nodes are the *A-Chain* and *AB-Chain* attributes, respectively. For the root node, the value of both attributes is zero. Similarly, the subscripts on the NP nodes represent the *node* and *Chain* attributes of the NP. The last NP in the tree, complement of *love*, bears *node* number 5, and belongs to chain 1; its position is Case-marked, so the NP is a Wh-trace. Chain 1 has two other elements, NP nodes 1 and 3, and is an A-Bar chain, since it is headed by the Wh-phrase at *node* 1, an A-Bar position.

---

20 The scope of the Chain rule in (19) and (20) is not as broad as that of move-α; it does not handle topicalizations, clefts, parasitic gaps, and multiple Wh-extraction from a given phrase. These topics are discussed below.

(21)



3.2.5 Correctness of the Chain rule

Well-formedness properties of chains, used in the formulation of the Chain rule, can now be shown to follow from the attribution that defines the rule.

3.2.5.1 C-command relation

The first property that we prove is the c-command relation between a trace and its antecedent. First, notice that both attributes *A-Chain* and *AB-Chain* are inherited (i.e., propagated from parents to children). From attribution rules (20.a) for the start production, CP.*A-Chain* = CP.*AB-Chain* = 0 for the matrix CP node. Where the value of *A-Chain* is zero, there is no possible NP-trace antecedent to the left of the node bearing the attribute; similarly, where *AB-Chain* is zero there is no possible antecedent for Wh-trace. Since the only rule which introduces an A-chain is in (20.c), for the IP production, and the I-specifier (Subject NP) c-commands the IB node at which the chain is introduced, and *A-Chain* is inherited, any NP-trace that may be assigned to this chain is c-commanded by the Subject antecedent.

Similarly, the only rule that introduces an A-Bar chain is in (20.b), for the CP production. This rule identifies the NP in C specifier position as the binding operator. Since C-specifier position c-commands CB, and *AB-Chain* is also inherited, it follows that Wh-traces are also c-commanded by the operator that binds them, and we may conclude the proof. By a slight extension of the preceding argument, it can be shown that every non-head (trace) element in a chain c-commands every trace in its right domain.

3.2.5.2 Subjacency

The next property we consider is Subjacency. This condition must obtain between adjacent chain elements. Suppose that $C=(\alpha_1, ..., \alpha_n)$ is an arbitrary chain, and let α_i and α_{i+1} be two adjacent elements. We show that these two elements must be subjacent by inspection of possible cases of movement. First, assume that C is an argument chain. Then α_i and α_{i+1} are in A-position. Assume α_{i+1} is the object of some verb V or preposition P, head of a complement of V.[21] Then, by attribution rules (20.d) and (20.e), α_{i+1} must be the last element of the chain and $\alpha_{i+1}.Chain = VB.A\text{-}Chain = VP.A\text{-}Chain$. By attribution rules (20.c), we also have $VP.A\text{-}Chain = IB.A\text{-}Chain = \gamma.Chain$, where γ is the Subject position of the clause of which V is main verb. Hence $\gamma.Chain = \alpha_{i+1}.Chain$ and γ is the NP position closest to α_{i+1}; it follows that $\gamma = \alpha_i$ and we may conclude that α_i and α_{i+1} are subjacent. The case we have just reviewed corresponds to movement of object to subject position. Note that $\gamma.\theta = nil$, so that γ is a θ-Bar position.

Next, suppose that α_{i+1} is Subject position of some sentence IP. Then, by attribution (19.a) and (20.c), and since by hypothesis α_{i+1} is an NP-trace, $\alpha_{i+1}.Chain = IP.A\text{-}Chain$. By attribution rules (20.a), $IP.A\text{-}Chain = CB.A\text{-}Chain = CP.A\text{-}Chain$, for the closest ancestor CP. Now, CP cannot be the matrix clause, for otherwise $CP.A\text{-}Chain = 0$, which leads to a violation of the Chain condition (19.b). Similarly, CP cannot be a relative clause complement in some noun phrase NP, for otherwise (20.e) leads to the same violation. Thus CP is the clausal complement of some verb V. By attribution rules (20.d), $CP.A\text{-}Chain = VB.A\text{-}Chain = VP.A\text{-}Chain$, and we fall into the

---

[21] In fact, α_{i+1} cannot be object of a preposition, since prepositions assign Case and α_{i+1} is NP-trace. In constructions like (i.a) the verb and preposition form a verb-preposition constituent; contrast with (i.b).

(i) a. [The boat]$_i$ was fired upon [ϵ]$_i$ by the army.
b. * Cambridge$_i$ was gone [to [ϵ]$_i$] by me.

previous case, which leads to $\gamma.Chain =$ IB.*A-Chain*= VP.*A-Chain*, where γ is the subject of the clause of V. Thus $\alpha_i = \gamma$, and α_i is subjacent to α_{i+1}, since they are separated by the single bounding node IP (for English).

If the chain C is a non-argument chain, subjacency can be proved to hold in a similar fashion. Briefly, the two subcases to be considered are, first, Wh-movement from object or subject position within a clause CP to C-specifier position of the same clause and, second, C-specifier to C-specifier movement, from CP to higher clauses. In both cases subjacency can be shown to hold, since the attribute *AB-Chain* is propagated by the attribution rules along Wh-chain hub nodes, requiring a Wh-trace in every intermediate C-specifier position. This kind of chain indexing corresponds to successive cyclic movement, as in (22.a). Wh-island phenomena are impossible, since once an A-Bar chain is assumed to propagate across a given CB node, a trace is required at the corresponding C-specifier, and the trace may be assigned to no other A-Bar chain. If the type of the *AB-Chain* attribute is changed to a stack of integers, as discussed in section 3.2.6 below, subjacency and/or strict cycle violations (22.b) are possible.

(22) a. Who$_i$ do [$_{IP}$ you think [$_{CP}$ [ϵ]$_i$ that John said [$_{CP}$ [ϵ]$_i$ that Bill should fire [ϵ]$_i$]]]

b. * Where$_i$ do [$_{IP}$ you think [$_{CP}$ [$_{NP}$ which books]$_j$ [$_{CP}$[ϵ]$_i$ Bill bought [ϵ]$_j$ [ϵ]$_i$]]]

Last we consider extraction from an embedded NP. While the topic is still controversial and the theoretical issues unresolved (van Riemsdijk and Williams (1986)), for English the basic fact is that extraction out of NP is illicit, as suggested by (23). This principle, known as the Complex Noun-Phrase Constraint, was one of the first constraints on transformations (Ross, 1967). In constructions where the principle is apparently violated, as in (24.a), the process of extraction is limited to movement to C-specifier from object position (24.b), and the possibility of extraction is an idiosynchratic property of certain verb and complement types (24.c-d).

(23) a. * What$_i$ does [$_{NP}$ the book about [ϵ]$_i$] cost ten dollars.

b. * What$_i$ did you buy [$_{NP}$ a book about [ϵ]$_i$]

c. * [$_{PP}$ About what]$_i$ did you buy [$_{NP}$ a book [ϵ]$_i$]

d. * Where$_i$ did [$_{NP}$the man [$_{CP}$ [ϵ]$_j$ [ϵ]$_i$ that [$_{IP}$ John shot [ϵ]$_j$ [ϵ]$_i$]]] die.

(24) a. Who_i are you going to take $[_{NP}$ a picture of $[\epsilon]_i$]

b. * Who_i was $[_{IP}$ a picture of $[\epsilon]_i$] taken.

c. * Who_i are you going to destroy $[_{NP}$ a picture of $[\epsilon]_i$]

d. * $What_i$ did Atila cause $[_{NP}$ the destruction of $[\epsilon]_i$]

Van Riemsdijk and Williams (1986) suggest that the grammaticality of (24.a) might be explained by assuming that the complement of the verb is not an NP, but instead NP followed by PP, as in (25.a). Wh-movement can then take place from the PP, in a fully productive manner (25.b). The problem of explanation is transfered to the lexicon, as it will now need to state that verbs like *take* may optionally subcategorize a PP_{of} complement.

(25) a. Who_i did John take $[_{NP}$ a picture] $[_{PP}$ of $[\epsilon]_i$]

b. Who_i did Mary give the book $[_{PP}$ to $[\epsilon]_i$]

The present formulation assumes the strict version of the Complex Noun-Phrase Constraint, by which no movement can take place out of NP. As can be seen from attribution rules (20.e), the attributes *A-Chain* and *AB-Chain* are not propagated across NP nodes; thus chains across NP nodes are impossible.

To conclude this section, we investigate the manner in which the subjacency facts are explained by the present attribute grammar implementation. Notice first that there is no particular set of categories in the theory that have been declared as Bounding categories. There is no special procedure that checks that the Subjacency condition is actually satisfied by, say, traversing paths between adjacent chain elements in a tree and counting bounding nodes. Instead, the facts are implied by the attribution that defines the values of the attributes *A-Chain* and *AB-Chain*. The detailed account above shows that NP-movement is from object or Subject position to the nearest Subject which c-commands the extraction site. Similarly, Wh-movement is from object, Subject, or C-specifier position to the nearest c-commanding C-specifier. Since *AB-Chain* is integer-valued, at most one A-Bar chain may propagate, and either IP or CB (but not both) may be taken as bounding nodes, and Wh-island phenomena are observable. NP is a bounding node

as a consequence of the strong condition that no chain spans across an NP node, which in turn implied by the rules (20.e).

3.2.6 Bounding nodes and Path containment condition (PCC)

If the Subjacency condition is not imposed on movement, it is posible for more than one A-Bar chain to propagate across a given bounding node, as illustrated in (26.a). It has long been recognized, however, that the double extraction structure obeys a last-in/first-out (lifo) constraint, as revealed by the far worse ungrammaticality of (26.b).[22]

(26) a. * [What books]$_i$ do you know who$_j$ to persuade [ϵ] $_j$ to read [ϵ]$_i$

b. ** Who$_i$ do you know [what books]$_j$ to persuade [ϵ] $_i$ to read [ϵ]$_j$

Pesetsky (1982) has proposed a constraint, the *Path Containment Condition* or PCC, from which the observed lifo nesting may be deduced. We may account for this phenomenon by assuming that the attribute *AB-Chain*, which propagates Wh-chains along hub nodes, is not an integer, but instead a lifo stack of integers, into which A-Bar chain numbers are pushed as their heads are first encountered. The lifo constraint would then require that chains be terminated in the reverse order in which heads were pushed. We do not implement this mechanism, though, since for English the constructions (26.a) are marginal at best. We maintain the Subjacency condition on movement, and keep the domain of the *AB-Chain* attribute as defined above.

The examples (26) are interesting, since equivalent constructions are grammatical in other languages. It is assumed in Government-binding that the set of bounding nodes that a language may select is not fixed across human languages, but is open to parametric variation. Rizzi (1978) observed that in Italian the Subjacency condition is systematically violated in double Wh-extraction constructions, as in (27.a). The analogous construction (27.b) is also possible in Spanish. One solution, considered by Rizzi to explain the grammaticality of (27), is to assume that in Italian and Spanish, C-specifier position may be "doubly filled" in the course of a transformational derivation, while requiring that it be not doubly filled by non-empty phrases at S-Structure. Thus both moved phrases *a cui* and *che storie* can move to the lowest C-specifier

---

[22] Examples and judgements are from van Riemsdijk and Williams (1986).

position in the first transformational cycle, while in the second cycle *a cui* may move to the next higher C and *che storie* stays in the first C. A second solution, which is adopted by Rizzi and constitutes the currently accepted explanation of the (apparent) Subjacency violation, is to assume that Italian and Spanish select CP and NP as bounding nodes, a set different from that of English. The first phrase *che storie* may then move to the lowest C position in the first transformational cycle, while the second, *a cui*, moves in the next cycle in one step to the next higher position, crossing two IP nodes but, crucially, only one CP node. Thus Subjacency is satisfied if CP, not IP, is taken as the bounding node.

(27) a. Tuo fratello, $[_{PP}$ a cui$]_i$ mi domando $[_{NP}$ che storie$]_j$ abbiano raccontato $[\epsilon]_j$ $[\epsilon]_i$, era molto preoccupato.
Your brother, to whom I wonder what stories they have told, was very worried.

b. Tu hermano, $[_{PP}$ a quien$]_i$ me pregunto $[_{NP}$ que historias$]_j$ le habran contado $[\epsilon]_j$ $[\epsilon]_i$, estaba muy preocupado.

The empirical data that arguably distinguishes between the two proposed solutions is (28.a). While the "doubly filled" C hypothesis allows indefinitely long Wh-chain hubs of doubly filled Cs, making it possible for a Wh-chain element and its successor to skip more than one C position that already contains some Wh-phrase, the "bounding node" hypothesis states that at most one filled C position may be skipped. Thus, the second hypothesis, but not the first, correctly predicts the ungrammaticality of (28.a).

(28) a. * Juan, $[_{PP}$ a quien$]_i$ no me imagino $[_{NP}$ cuanta gente$]_j$ $[\epsilon]_j$ sabe donde$_k$ han mandado $[\epsilon]_i$ $[\epsilon]_k$, desaparecio ayer.
Juan, whom I can't imagine how many people know where they have sent, disappeared yesterday.

b. La Gorgona, $[_{PP}$ a donde$]_i$ no me imagino $[_{NP}$ cuanta gente$]_j$ $[\epsilon]_j$ sabe $[_{PP}$ a quienes$]_k$ han mandado $[\epsilon]_k$ $[\epsilon]_i$, es una bella isla.
La Gorgona, to where I can't imagine how many people know whom they have sent, is a beautiful island.

One may observe that (28.a), even if it satisfies subjacency, violates Pesetsky's (1982) Path Containment Condition (PCC). On these grounds, (18.a) does not decide between the two

hypotheses. (28.b), on the other hand, which is structurally similar to (28.a) but satisfies the PCC, argues in favor of the "doubly filled" C hypothesis. The Wh-phrase *a donde* moves from its D-Structure position to the surface position, skipping two intermediate C positions. This is possible if we assume the doubly filled C hypothesis, and would violate Subjacency under the alternate hypothesis, even if CP is taken as the bounding node. We expect a similar pattern (28.b) to be also valid in Italian.

Movement across doubly filled C nodes, satisfying Pesetsky's (1982) Path Containment Condition, may be explained computationally if we assume that the *type* of the *AB-Chain* attribute on chain hub nodes is a last-in/first-out (lifo) stack of integers, into which the integers identifying A-Bar chain heads are pushed as they are first encountered, and from which chain identifiers are dropped as the chains are terminated. Only the chain denoted by the top stack element has "access" to the trace in complementizer position. If we further assume that the type of the attribute is universal, we may explain the typological difference between Italian and English, as it refers to the Subjacency condition, by assuming the presence of an A-Bar chain *stack depth bound*, which is parametrized by universal grammar, and has the values 1 for English, and 2 (or possibly more) for Italian and Spanish. When this bound is 2 or greater, CP is the closest approximation to a bounding node (although cf. (28.b)), and Wh-island violations which satisfy the PCC are possible.

3.2.7 Topicalization and clefting

A slight extension of the Chain rule above permits accounting for topicalized and cleft constructions, as in (29).

(29) a. Bagels$_i$, I know he likes [ϵ]$_i$.

b. It is bagels$_i$ [$_{CP}$ that he likes [ϵ]$_i$]

The empty category at the extraction site is governed and Case-marked, and hence must be a Wh-trace. The relation between adjacent pairs of traces in the movements is subjacent. The type of movement that occurs in topicalization and clefting is then Wh-movement (van

Riemsdijk and Williams, 1986). However, the target position for movement is not C-specifier position, but rather Chomsky-adjoined to CP, as in the more detailed rendering (30) of (29.a).[23]

(30) a. $[_{CP}$ Bagels$_i$, $[_{CP}$ $[\epsilon]_i$ I know $[_{CP}$ $[\epsilon]_i$ he likes $[\epsilon]_i$]]].

The Chomsky-adjoined structure cannot be generated by the base phrase structure rules we have assumed so far. We assume the additional rule (31), which generates the topic position XP, sister of CP; the category XP ranges over NP, PP, CP, and several others. The rule is violates succession in the X-Bar convention. Topic position is a non-argument position -- no θ-role is assigned directly there. The attribution associated with (31) extend the Chain rule, and properly relate a topicalized phrase to intermediate traces and the extraction site.

(31) $CP_1 \rightarrow XP\ \ CP_2$
attribution:
$CP_2.AB\text{-}Chain \leftarrow XP.Chain$
condition:
$XP.empty = '-'$
$CP_1.x = 0$, for $x = A\text{-}Chain, AB\text{-}Chain$

The topicalization rule (31) requires overt topics, and permits topicalization in embedded clauses, as in the topicalized form (32.b) of (32.a); the target site of the topicalized phrase need not be adjoined to the matrix CP node. Note that the attribute conditions in (31) do not permit NP- or Wh-movement outside a topicalized CP node; this explains (32.c-d). Topicalization is an instance of Wh-movement; it does not limit NP-movement across a clause with a topicalized constituent (32.e), as long as the movement is not to a position outside the node dominating the topic position.

(32) a. It seems everybody likes bagels.

b. It seems $[_{CP}$ bagels$_i$ everybody likes $[\epsilon]_i$]

c. * everybody$_j$ seems $[_{CP}$ bagels$_i$ $[\epsilon]_j$ to like $[\epsilon]_i$]

---

[23] Chomsky (1977) assumes that topic position is base-generated by a rule like CP′ → TOPIC CP.

d. * Who_j does $[_{IP}$ it seem $[_{CP}$ $bagels_i$ $[\epsilon]_j$ likes $[\epsilon]_i$]

e. $Bagels_i$ $everybody_j$ seems $[_{CP}$ $[\epsilon]_j$ to like $[\epsilon]_i$]

We omit the formulation of the attribution rules that account for clefts. The mechanism involved is similar to that of topicalization.

3.3 Government, Case-marking, and θ-marking

The subtheories that define the notions of government, Case, and thematic role assignment explain a large number of grammaticality questions in constructions. In this section we review the manner in which the government relation and the processes of Case- and Theta-marking are defined. The well-formedness conditions that apply to the operation of the three mechanisms (ECP, Case filter, Theta Criterion, and Projection Principle) are discussed in Chapter 2, and it may be shown that they are satisfied by the Chain rule.

3.3.1 Government

Government is a binary relation formally defined in (76) in Chapter 2. It is repeated here as (33). The structural configuration described is shown in (34), where XP denotes the maximal projection of X, and the path between XP and Y, excluding the end-nodes, does not contain any maximal projection.

(33) α *governs* β if and only if

(i) α is a zero-level category (if $\alpha = I$, then *AGR* is present), and

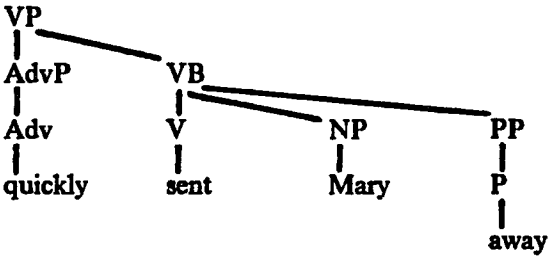
(ii) α m-commands β, and

(iii) each maximal projection γ dominating β also dominates α.

(34)

Our interest in the government relation lies in the role it plays in Case- and θ-role assignment and the distribution of empty categories. By definition (33), only the head X of a maximal projection XP may be governor of other nodes under XP. Complements and specifiers of X are maximal projections, assuming X-Bar theory. Hence, they are not potential governors. Any other node Y dominated by XP has a maximal projection YP intervening between Y and XP, and cannot be a governor for nodes outside YP. Similarly, Y cannot be governed by a node outside YP. The fact that only X may govern its complements and specifiers is due to the use of the m-command relation in (33.ii). If c-command were used instead, several nodes could be governors of another one, as noted in (35). The node NP in (35) is governed (in the c-command sense) by nodes other than V, including Adv, and P; of these only V is the relevant governor.

(35)



By the maximality condition of X-Bar theory, only maximal projections may appear as specifiers or complements of the head X in (34). It follows from clause (33.iii) that only maximal projections may be governed. In light of the remarks of the preceding paragraph, it is possible to reduce the notion of government within a maximal projection XP to a (binary) relation that may obtain between the head X and its specifiers and complements. For the present attribute-grammar specification, we assume an attribute *Gov*, which is defined for each maximal projection XP. The domain of the attribute is ordered pairs $<i, prodrop>$, where *i* is the *node* number of XP's governor, if it has one, or zero, if it has none, and *prodrop* is a label *proper* or *nil* that indicates whether the governor, if any, is proper. Furthermore, an attribute *head* is defined for every single-bar and maximal projection, and its value is the *node* number of the head of the projection. The value of the *prodrop* parameter is either *proper* or *nil*, and set by universal grammar (Chomsky, 1981; Taraldsen, 1978). For English and French the value is proper for all categories, except I, for which the value is *nil*. Italian and Spanish the value is *proper* for all

categories. The attribution rules that implement the government mechanism described are shown in (36.a) for single-bar level categories, and in (36.b) for maximal projections.

(36) a. $XB \rightarrow X \quad YP_1 \ldots YP_n$
attribution:
$YP_i.Gov \leftarrow \langle X.node, prodrop \rangle$, for $i = 1, \ldots, n$.
$XB.head \leftarrow X.node$

b. $XP \rightarrow YP_1 \ldots YP_n \quad XB$
attribution:
$YP_i.Gov \leftarrow \langle XB.head, prodrop \rangle$, for $i = 1, \ldots, n$.
$XP.head \leftarrow XB.head$

Certain verbs, such as *seem* and *believe* in (37) can exceptionally govern the specifier position of their clausal complement, as noted in Chapter 2. In (37.a), the empty category $[\epsilon]_i$ is left by NP-movement. The category is subject to the ECP, so it must be properly governed. Since the clause CP in which it occurs is infinitival, its governor is not to be found inside CP, but outside. Under our current assumptions, this is impossible, since both CP and IP are maximal projections and thus barriers to government. The solution noted in Chapter 2 assumes that a verb may subcategorize for a clausal complement CP in which the CP and IP maximal projections are "transparent" to government (Chomsky, 1981) -- i.e., they do not count as maximal projections for clause (33.iii) in the definition of government. Thus the matrix verb *seem* may properly govern the embedded empty subject. Similarly, in (37.b), the subject *John* of the complement is exceptionally Case-marked by the matrix verb *believe*. Since (structural) Case-marking takes place only under government, it follows that *believe* also exceptionally governs *John*.

(37) a. $John_i$ seems $[_{CP}[_{IP}\ [\epsilon]_i$ to be here]]

b. I believe $[_{CP}[_{IP}$ John to be here]]

The attribution rules responsible for *exceptional government* appear in (38). The attribute *ecm* is binary and indicates whether a clausal complement permits exceptional government, as noted above. The attribute is an inherent lexical property of verbs, and is positively specified for verbs like *believe* and *seem*. The value of the attribute may be assigned by the verb to the root node CP of its clausal complement. This part of the attribution is not shown in (38). The attribute

interacts with the *forcomp* and *tense* attributes of C and IB, which record, respectively, the type of complementizer heading CP, and whether I is tensed. Thus, if I is tensed, it governs its Subject NP, and does so properly or not, according to the value of the *prodrop* parameter in the attribution rule. In Italian and Spanish a tensed I may properly govern its Subject position and small "pro" Subjects are allowed. If I is untensed, its specifier may still be governed (and Case-marked) by the complementizer preceding it, if present. If the complementizer is not present and the clause is marked *ecm*, the I specifier will be governed by the clause's governor, recorded in the attribute IP.*Gov*.

(38) a. CP → (XP) CB
attribution:
CB.x ← CP.x , for x = *ecm*, *Gov*

b. CB → C IP
attribution:
IP.*Gov* ← if C.*forcomp* = '−' then <C.*node*, *proper*>
else if CB.*ecm* = '+' then CB.*Gov* else <0, *nil*>

c. IP → NP IB
attribution:
NP.*Gov* ← if IB.*tense* = '+' then <IB.*head*, *prodrop*> else IP.*Gov*

This concludes our cursory account of government.[24]

3.3.2 Case-marking

Noun phrases in a sentence are assigned a grammatical Case in virtue of their position in the sentence. Once Case is assigned to a noun phrase, inflectional properties of the language determine whether and in which manner the Case is morphologically realized by the phrase. The fundamental instances of Case-assignment reviewed in Chapter 2 are in (91); we repeat them here as (39) (Chomsky, 1981).

---

[24] We have omited the case of proper government by a local antecedent.

(39) a. NP is *nominative* if governed by a tensed I.

b. NP is *objective* if governed by V with subcategorization frame [NP ...]

c. NP is *oblique* if governed by P.

d. NP is *genitive* if it appears in NP specifier position ($[_{NP}$ _ ...])

e. NP is inherently Case-marked by V in double-object constructions.

Case-assignment is defined by the attribute grammar as follows. An inherent attribute *Case* is defined for V and P. The domain of this attribute is (40), and its value is determined by lexical properties of the item inserted under the V or P node. The attribute *Case* is also associated with NP, but now as an inherited attribute. For Case-assignment instances (39.b-c), the attribute is assigned under adjacency by the V or P Case-assigner to the NP, as indicated by the attribution rules (41.a-b). Inherent Case-marking (39.e) is achieved by assuming that *Case* propagates across the first object to the second, as in (41.c). Thus in double-object constructions both objects bear the same Case. In languages like German, the verb pressumably has several Cases associated with it, which differ for the first and second objects.

(40) *Nom, Acc, Dat, Oblq, Gen, nil*

(41) a. VB → V NP (PP*) (CP)
attribution:
NP.*Case* ← V.*Case*

b. PB → P NP
attribution:
NP.*Case* ← P.*Case*

c. VB → V NP_1 NP_2
attribution:
NP_i.*Case* ← V.*Case*, for $i = 1, 2$

d. IP → NP IB
attribution:
NP.*Case* ← if IB.*tense*='+' then *Nom* else IP.*Case*

e. NP → NP_1 NB
attribution:
NP_1.*Case* ← *Gen*

Nominative Case is assigned in I specifier position under government by a tensed I (41.d). As for exceptional government, if the I node is untensed, its NP specifier may still be Case-marked exceptionally by the clause complementizer, or by the verb which governs the clause. The attribution rules relevant for exceptional Case-marking are shown in (42) and are parallel to those for exceptional government. Notice that the *Case* attribute is associated with CP and IP as an inherited attribute, and that its value propagates to the Subject NP if the *ecm* conditions are satisfied. The *Case* value on the CP node is *nil* for the root node and is otherwise determined by the position of the node. For example, if CP is complement of a verb, the verb Case-marks the node. This part of the attribution is not shown in (42). The last instance of Case-assignment (39.d) is implemented in a straightforward way by attribution rule (41.e).

(42) a. CP → (XP) CB
attribution:
CB.*Case* ← CP.*Case*

b. CB → C IP
attribution:
IP.*Case* ← if C.*forcomp*='+' then *Dat*
else if C.*empty*='−' then *nil*
else if CB.*ecm*='+' then CB.*Case* else *nil*

c. IP → NP IB
attribution:
NP.*Case* ← if IB.*tense*='+' then *Nom* else IP.*Case*

The attribute *Case* is also associated with the categories NB and N, as an inherited attribute. Lexical insertion of a head noun under the category N requires match of the *Case* value on the two symbols.

3.3.3 Theta-marking

Predicate-argument structure in a sentence is defined by means of thematic roles (θ-roles) and the process that assigns them. The elements of thematic theory that define this process are discussed in Chapter 2, section 2.7.

For the present attribute grammar implementation we assume two attributes θ_E and Θ_I associated with each lexical element that may assign a θ-role. The attributes are thus associated with V and P. Attribute θ_E records the name of the external theta-role assigned by the predicate and has the value domain (43.a). Θ_I has type (43.b), and is a list of the names of the θ-roles that the predicate assigns internally. We assume that there is a one-to-one relation between the elements of Θ_I and the syntactic categories in the attribute *subcat*, the strict subcategorization frame of the predicate. If θ_i and XP_i are the *i*-th elements of Θ_I and *subcat*, respectively, the grammatical function in which θ_i is assigned is $[XP_i, VB]$.

(43) a. $\Theta = \{$ *agent, theme, goal, goal, source, place, means, situation, nil* $\}$

b. $\{ [\theta_1, ..., \theta_n]: n \geq 0 \text{ and } \theta_i \in \Theta \}$

For each syntactic category that may appear as argument of a predicate, we assume an inherited attribute θ, which records the name of the θ-role assigned to that argument by the predicate.[25] Following X-Bar theory assumptions, only maximal projections may be arguments. The attribute θ is associated with NP, PP, AP, AdvP, and CP. The type of θ is (43.a). Internal theta-roles areassigned under government by attribution rules (44.a-b). Notice how the θ-role of objects of prepositions is assigned by the governing preposition, as in (44.b). One might assume that the θ-role assigned by the preposition, in turn, is determined compositionally by it and the verb which θ-marks the PP projection, as suggested by Stowell (1981). However, our present

---

[25] θ-role assignment in the approach to θ-roles in (Kornfilt and Correa, in preparation) is binding the semantic reading of the argument into the logical form of the predicate.

understanding of θ-roles does not call for the fine distinctions that this extra mechanism would afford, and the present implementation reflects the assumption that the θ-role assigned is determined by the preposition alone.

(44) a. $VB \rightarrow V\ XP_1, ..., XP_n$
attribution:
$XP_i.\theta \leftarrow V.\Theta_I(i)$, for $i = 1, ..., n$.

b. $PB \rightarrow P\ NP$
attribution:
$NP.\theta \leftarrow P.\Theta_I(1)$

The external argument of a predicate (verbs alone) is θ-marked compositionally by the verb and the first IB projection that dominates the verb (Chomsky, 1981). The external θ-role is first propagated upwards, to the IB projection, as in (45.a-c), and then assigned under government by IB (45.d). Notice that an external θ-role is always assigned, but its value may be *nil*, which we take to be an empty set of thematic relations. Since the θ-role must be assigned, an I specifier must be present in every clause, and we obtain the part of the Extended Projection Principle which requires a Subject in every clause. The remainder is discussed in Chapter 2, in section 2.7 on thematic theory.

(45) a. $VB \rightarrow V$ ...
attribution:
$VB.\theta_E \leftarrow V.\theta_E$

b. $VP \rightarrow$ ... VB ...
attribution:
$VP.\theta_E \leftarrow VB.\theta_E$

c. $IB \rightarrow I\ VP$
attribution:
$IB.\theta_E \leftarrow VP.\theta_E$

d. IB → I VP
attribution:
NP.θ ← IB.θ_E

In addition to the above cases of θ-role assignment, some phrases may be θ-marked by virtue of the configuration in which they appear, in manner analogous to Case-marking in configuration (41.d). An NP in the context [$_{NP}$ _ ...] may be assigned the θ-role *place*, as in (46.a), unless the head noun assigns a different role. Depending on the treatment adopted of modifier phrases, one might resort to this extended mechanism for theta-marking, as in (46.b-c).

(46) a.NP → NP_1 NB
attribution:
NP_1.θ ← if NB.θ_E=*nil* then *place* else NB.θ_E

b. VP → VB CP
attribution:
CP.θ ← *purp, manner,* etc.

c. VP → VB PP
attribution:
PP.θ ← *place, means,* etc.

3.4 Structural Determination of Empty Categories

The empty categories we assume are NP-trace, Wh-trace, and PRO. All three arise by expansion of an NP node using the ϵ-production (47).

(47) NP → ϵ

The distribution and functional classification of empty categories parallels the distribution of overt noun phrases. The attributes *pronominal* and *anaphoric* yield a functional classification of empty and overt NP types as in (47), Chapter 2, repeated as (48).

(48)

	Referential (Wh-trace)	**Anaphor (NP-trace)**	**Pronominal (pro)**	**-- (PRO)**
anaphoric	−	+	−	+
pronominal	−	−	+	+
empty	− (+)	− (+)	− (+)	− (+)

The functional type of an overt phrase is determined from the lexical material contained in the phrase. For an empty category the functional type is determined as discussed in Chapter 2, section 2.6.4, without reference to the antecedent that the category may have. Instead we look at the argument/non-argument status of the position it occupies, and whether the position is Case-marked and governed. The classification (113) of Chapter 2, is repeated as (49) for convenience.

(49)

	A-position	**Government**	**Case**
Wh-trace (variable)	+	+	+
Wh-trace (COMP)	−	+	−
NP-trace	+	+	−
PRO	+	−	−

The attribution rules (50) responsible for determining the functional type of an NP are simply a particular translation of table (49). We let the attributes *anaphoric* and *pronominal* be synthesized, and assume a binary-valued attribute *A-Bar*, which records whether the NP is in an argument or non-argument position. Note that the Chain rule (19) is data-dependent on the attributes *anaphoric* and *pronominal*, so that the attribution (50) must apply before the Chain rule applies. Similarly, the attribution (50) is data-dependent on the attributes *Case*, *Gov*, and *empty* of NP; the values of these attributes must be determined before functional type can be established.

(50) NP → ∈ | ... NB ...

attribution:

NP.*anaphoric* ← if NP.*empty*='−' then NB.*anaphoric*
else if NP.*A-Bar*='+' then '−'
else if NP.*Gov*= <0, *nil*> then '+'
else if NP.*Case*=*nil* then '+' else '−'

NP.*pronominal* ← if NP.*empty*='−' then NB.*pronominal*
else if NP.*Gov*= <0, *nil*> then '+' else '−'

3.5 English Auxiliary System, without Affix-hopping

It is still a debated issue what the categorial status and defining properties of the auxiliary are. In spite of this, it is generally agreed that the auxiliary elements in a sentence are those marking tense, aspect, and/or modality (Steele, 1981). Chomsky (1957) proposed the context-free rule (51) to account for the order of auxiliary elements in English. This rule interacts with a transformational rule of *affix-hopping*, to derive the correct attachment of affixes to verbal elements.

(51) AUX → C (M) (have *-en*) (be *-ing*) (be *-en*)

The element C in (51) may be rewritten into agreement and tense morphemes (affixes), observing certain contextual conditions. The morphemes *-en* and *-ing* are base-generated and indicate aspectual properties that the element preceeding requires of the next verbal element. The morpheme *-en* introduced by *have* indicates perfect aspect, and is affixed to the verb following *have* to form a past participle. The second affix, *-ing*, marks progressive aspect, and attaches to the verb following *be* to form a present participle. The third morpheme, *-en*, marks the passive voice and affixes to a verb to yield a passive form. Examples of the use of the three affixes is given in (52).

(52) a. John has eaten.

b. John is eating.

c. John was eaten.

In Chomsky's account, base-generation of an affix is followed by application of the affix-hopping transformation, to move and attach the affix to the next verbal element in the sentence. Structure (53.a) may be converted to the surface form (53.b) after affix-hopping applies.

(53) a. They [may have-*en* be-*ing* be-*en*] beat.

b. They [may have be-*en* be-*ing*] beat-*en*.

The affix-hopping transformation was originally offered as an illustration of the power of transformational grammar. The rule, however, has been criticised at various time for a number of technical difficulties it brings to the transformational component of the grammar (Akmajian and Wasow, 1975). These are reviewed in Gazdar, Pullum and Sag (1982). In the present section an alternate account of affix-hopping is developed which does not rely on transformation, but instead on attribute propagation in the AUX subtree. The approach is similar to that of Gazdar, Pullum and Sag, and to that of Farmer (1984), in that it uses a form of subcategorization to require the correct attachment of affixes. It differs from theirs in that we assume the flat constituent analysis of AUX proposed by Steele (1981) and in the use of both inherited and synthesized attributes to achieve the affix propagation. While we do not intend to go into the complicated linguistic issues surrounding the AUX controversy, the implication of the grammar fragment presented here is that neither of the newer two accounts is to be preferred over Steele's, on the grounds that they do not use affix-hopping. Independent linguistic factors will need to be considered to decide what the proper constituent relations of auxiliary elements is.

3.5.1 The AUX controversy

There are two basic hypotheses about the English auxiliary system. The auxiliary rule (51) implies that AUX is a syntactic category of English, which exhaustively dominates at D-Structure any lexical items denoting *modality*, *aspect*, and, if any of the previous two are present, also *tense*. If neither modality or aspect is present, *tense* is affixed to the main verb of the clause. Given the set of category labels we assume, we let the non-lexical category I denote AUX; however, in the present section we use the label AUX rather than I, to focus attention on the auxiliary. Rule (51) is supplemented by rules (54.a-b), which give the structural position of the auxiliary node within a sentence. The configuration that obtains is (55).

(54) a. IP → NP IB

b. IB → AUX VP

(55)

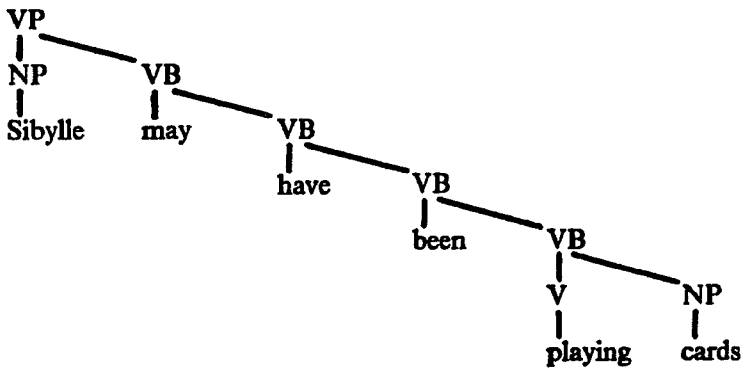
The analysis of AUX that (51) and (54) yield is close to that of Chomsky (1957) and Steele (1981). Crucially, it defines AUX as a syntactic category of English, distinct from verbs, and assumes that the auxiliary elements in a sentence form a constituent (at D-Structure). Rule (51) is outside the X-Bar convention. The AUX subtree is headless and none of the auxiliary elements takes complements, in the sense of X-Bar theory. VP is complement of the AUX node. AUX is not a lexical category, but it may dominate several lexical categories, including M. Steele (1981) presents an informal sketch of how the interpretation of IP phrases containing auxiliary elements may be obtained.

Although the analysis of AUX presented above has prevailed in much work on generative grammar, a number of objections have been raised against it. The most serious of them, perhaps, is the denial of the existence of the category AUX. Ross (1967b), and more recently Gazdar, Pullum, and Sag (1982), and Farmer (1984), deny the existence of the category in question, and claim instead that purported auxiliary elements are simply instances of main verbs. In Ross' account these main verbs take clausal complements, while in Gazdar's they take VB complements. Farmer uses basically the same approach of Gazdar, Pullum and Sag. In Gazdar's Generalized Phrase Structure Grammar, auxiliary elements do not belong to a single category, but instead are divided among several intersecting verbal subcategories, depending on the type of VB complement that they take. The relevant production schema in Gazdar, Pullum, and Sag's account are those in (56). The phrase-structure tree that these productions define for a sample sentence is shown in (57).

(56) a. $VP \rightarrow NP\ \ VB$ (where VP = IP)

b. $VB \rightarrow V\ \ VB$

(57) Sibylle may have been playing cards.



Steele (1981) provides a cross-linguistic study of the auxiliary which supports the hypothesis of AUX as an identifiable category distinct from V. She, furthermore, questions the validity of analyses such as (57), which assign a high degree of structuring to auxiliary elements.[26] In particular, Steele claims that in Luiseño (an American-Indian language) there is support for the claim that the auxiliary particle sequence is a constituent, unlike the claim implicit in (57). Steele (1981) lists the four non-definitional properties (58) of the auxiliary systems of (a small set of studied) natural languages.

(58) (i) Every instantiation of AUX in a language yields a *fixed* order for the auxiliary elements. For English the internal order is (M)(*have*)(be_1)(be_2), where be_1 introduces progressive aspect and be_2 introduces the passive.

(ii) If Subject-auxiliary inversion has taken place, the sequence of auxiliary elements, ignoring the presence of the subject NP, obeys the order noted in (i).

(iii) The relative internal order of elements in AUX does not follow from any known linguistic principles. The linear order does not reveal or is determined by any semantic relationships; within AUX there are no "main verbs" or "higher predicates."

(iv) The structure of AUX is outside the X-Bar convention. AUX is a headless category, in the sense of X-Bar theory.

---

[26] Steele's claim is that (57) is not warranted as a linguistic description by standard arguments about constituency. See also section 2.3.4 in Chapter 2.

As we have already remarked, we do not intend to enter into the AUX controversy, but only present the main issues involved. We adopt for the development below the analysis of the auxiliary given by rules (51) and (54) -- i.e., an analysis close to Chomsky's and Steele's version. The motivation we have for this choice is to show that affix-hopping is not crucial to support that analysis, and thus that the analysis by Chomsky and Steele is compatible with an attributed theory of grammar.

3.5.2 Specification of AUX

We identify AUX with the inflectional category I as a syntactic category of English, and possibly of other languages. AUX is distinct from V (main verbs); we may assume that AUX is [−N, +V, +*aux*], so it differs from V by the presence of the attribute [+*aux*]. The identification of AUX with I makes AUX non-lexical. The internal structure of AUX is determined by the production (58), where M, HAVE, BE1, BE2, and DO are lexical categories whose membership sets are indicated in (59); each of these categories is +*aux*. The optional specified formative *to* marks infinitival clauses and is ignored henceforth. BE1 and BE2 have identical extensions in English; in Spanish, however, the two extensions are different and correspond to the verbs *estar* and *ser*, respectively.

(58) AUX → M HAVE (*to*) BE1 BE2 | DO

(59)

M	may, can, will, ...
HAVE	have, had, ...
BE1, BE2	be, was, ...
DO	do, did, ...

Production (58) is similar to Chomsky's (1957) rule for AUX. An important difference between the two, however, is that affixes are not generated by the rule, as in (51) above. Here we assume that affixes are attached to other lexical forms at the lexical level. Word formation rules operate at a level below the syntax (or D-Structure), while they may change several syntactically relevant features of the words derived, including categorial features. This mechanism is independently used in the morphological component, for example to derive adjectival passives from verbs, or to suppress Case and external θ-role assignment by a passive verb. Both presuppose attachment of the affix *-en* to the verb. Note that if we assume affix-hopping takes place in the phonetic form (PF) component, as Chomsky (1981) assumes for non-pro-drop languages, we must entertain a complication in the derivation of syntactic passives; namely, affixation of the passive morpheme *-en* at PF entails or requires suppresion of *Case* and external θ-role at D-Structure. It is not

entirely clear how the relation between the feature changes at D-Structure and affixation of the passive morpheme at PF is to be established. Chomsky (1981) leaves the question open.

We assume three binary inherent attributes *part*, *perf*, and *pass*, which encode aspectual properties of auxiliaries and main verbs, and whether they are passive forms. In particular, the presence of the affixes *-en* and *-ing* that may be lexically attached to a verb implies the attribute values [+*part*, +*perf*] and [+*part*, −*perf*], respectively. At the syntactic level, an item like *been* is atomic (not formed by syntactic affixation of *-en* to *be*) and has morphological attributes [+*part*, +*perf*, −*pass*]. The categories M, HAVE, BE1, and BE2 in (58) bear inherited morphological attributes *part*, *perf*, and *pass*, and three synthesized counterparts $part_m$, $perf_m$, and $pass_m$, which are used by the lexical insertion rules to constrain the aspectual form of the items that may be inserted under the categories. Lexical insertion of an item under a node is permitted if the values of the attributes *part*, *perf*, and *pass* of the node match those of the lexical item. The synthesized attributes at the node depend on whether lexical insertion takes place under the node, and determine the aspectual properties of the next verbal element in the sentence. The basic mechanism applied (in place of affix-hopping) is thus subcategorization, as in (Gazdar, Pullum, and Sag, 1982). The attribution associated with the productions denoted by (58) appears in (60).[27]

(60) a. AUX → M HAVE BE1 BE2

attribution:

M.x ← AUX.x,

HAVE.x ← M.x_m,

BE1.x ← HAVE.x_m,

BE2.x ← BE1.x_m,

AUX.x_m ← BE2.x_m, for x = *AGR*, *mood*, *part*, *perf*, *pass*, *tense*

b. AUX → DO

attribution:

DO.x ← AUX.x,

AUX.x_m ← DO.x_m, for x = *AGR*, *mood*, *part*, *perf*, *pass*, *tense*

---

[27] I thank Prof. Kuno for pointing out an earlier error in the attribution

The values of the attribute occurrences *part*, *perf*, and *pass* at the root of an AUX subtree are inherited (or "input") and propagate horizontally through the subtree, through each intermediate auxiliary node, and emerge on the AUX node again as the synthesized attribute occurrences $part_m$, $perf_m$, and $pass_m$. The handling by the grammar of tense and agreement is similar. We do not posit free tense or agreement morphemes in the syntax, such as *past* or *-3rdS*, that could be used to derive (after syntactic affix hopping) the forms *had* or *has* from *have+past* and *have+3rdS*. Instead we assume the attributes *AGR* and *tense* in (60), which propagate in a similar manner to the other attributes and are used to obtain subject-verb agreement and enforce uniqueness of the *tense* feature in a clause. These attributes, as well as *mood* and the synthesized counterparts AGR_m, $mood_m$, and $tense_m$ are used below when we consider Subject-auxiliary inversion and Subject-verb agreement.

The auxiliary rule (58) departs from (51) also in that the categories on the right-hand side are not optional. This is only a "concrete syntax" detail, which reduces the number of productions we had to consider in the attribution (60). Lexical insertion under the categories M, HAVE, BE1, or BE2 is optional; this yields the same effect as the "abstract syntax" (51). The value of the attributes $part_m$, $perf_m$, and $pass_m$ at a node labelled by one of the categories, depends on whether lexical insertion takes place under the node.[28] The attribution (60) copies the values of the synthesized attribute occurrences at the node to the attribute occurrences *part*, *perf*, and *pass* at the next verbal element in the tree. As noted above, insertion of a lexical item under an auxiliary node creates a subcategorization context and corresponding lexical insertion condition for the next verbal element.

The lexical insertion rules for the categories M, HAVE, BE1, BE2, and DO, together with the conditions on their application are shown in (61). The attribution is summarized by means of the table (62). In (61.a) a lexical formative x is actually inserted under one of the auxiliary categories. Insertion of the item defines the value of the tuple [$part_m$, $perf_m$, $pass_m$] of synthesized attributes, as given by table (62). In (61.b) no item is inserted under the auxiliary category. This is done by expansion of the auxiliary category into the empty string 'ϵ'. The attribution associated with the rule simply propagates the values of the input auxiliary attributes [*part*, *perf*, *pass*] into the synthesized attributes [$part_m$, $perf_m$, $pass_m$] .

---

[28] The use of the concrete syntax (58) has the effect of increasing the number of attributes needed in the attribution (60). The synthesized attributes associated with auxiliary categories could be eliminated if the attribution were defined directly in the 32 CF productions defined by the abstract syntax (51).

(61) a. $X \rightarrow x$, for X = M, HAVE, BE1, BE2, DO

attribution:

$X.AGR_m \leftarrow nil$

$X.mood_m \leftarrow \alpha$

$X.part_m \leftarrow \beta$

$X.perf_m \leftarrow \gamma$

$X.pass_m \leftarrow \delta$

$X.tense_m \leftarrow -$

condition:

$[x.AGR, x.part, x.perf, x.pass, x.tense] = [X.AGR, X.part, X.perf, X.pass, X.tense]$

b. $X \rightarrow \epsilon$, for X = M, HAVE, BE1, BE2, DO

attribution:

$X.x_m \leftarrow X.x$, for $x = AGR, mood, part, perf, pass, tense$

(62)

X	α	β	γ	δ
M	+	−	−	−
HAVE	−	+	+	−
BE1	−	+	−	−
BE2	−	+	+	+
DO	−	−	−	−

By the attribution (61), if a lexical item is inserted under, say, HAVE, the synthesized attribute occurrences are $[+part_m, +perf_m, -pass_m]$. These values are then copied to the attribute occurrences [*part, perf, pass*] of the next auxiliary category, by the attribution rules of (60). The latter three attributes, in turn, create a lexical insertion condition for the element under it; namely, it must be $[+part, +perf, -pass]$. Thus, lexical insertion under HAVE requires that the next verbal element be a past participle, which is implied by affixation of *-en* to the element. For example, the lexical item *'beaten'* satisfies this condition since it is $[+part, +perf, -pass]$.[29] Similarly, if a lexical item is inserted under BE1, the attribution rules and conditions require that the next verbal element be $[+part, -perf, -pass]$, or a present participle. This is implied by affixation of *-ing* to the base verb. The lexical form *'beating'*, for example, satisfies the condition.

---

[29] The item *'beaten'* is actually generated ambiguously in the lexicon as a past participle $[+part, +perf, -pass]$ and as a passive participle $[+part, +perf, +pass]$. Thus, the item also satisfies the insertion condition (on the main verb) created by insertion under BE2. Which of the two ambiguous forms is used in a given context is determined by the context.

Notice that the empirical effect achieved is equivalent to affix-hopping, but no transformation is involved in the transfer of the feature complexes to the lexical item that requires them.

The position of the auxiliary node in a sentence is determined by the productions (54). The attribution associated with the productions is (63).

(63) a. IP → NP IB
attribution:
IP.AGR_m ← NP.*AGR*
IB.x ← IP.x, for x= *AGR, aux, mood, part, perf, pass, tense*

b. IB → AUX VP
attribution:
AUX.x ← IB.x, for x= *AGR, aux, mood, part, perf, pass, tense*
VP.x ← AUX.x_m, for x= *AGR, mood, part, perf, pass, tense*

The attributes *AGR, mood, part, perf, pass,* and *tense* are inherited when associated with IP, IB, and VP. The attribute AGR_m is synthesized by IP. The agreement attribute *AGR* is also associated with NP, as a synthesized attribute whose value is determined by the head noun. The attribution in (63) interacts with the attribution for Subject-auxiliary inversion, as discussed in the next section, to yield the correct distribution of auxiliary elements and morphological agreement between the Subject and the first verbal element in a sentence.

3.5.3 Subject-auxiliary inversion

In yes/no questions and certain other interrogatives, an auxiliary element may appear in pre-sentence position, under the complementizer C. This process is called Subject-auxiliary inversion. The current Government-binding account (Chomsky, 1986b) involves transformational movement of the first auxiliary element to complementizer position. The inversion is possible only in the matrix (topmost) clause. Here we assume that an auxiliary element may be base-generated in the C position of the matrix clause, by application of a

production in (64).[30] The element generated is related by attribution to the remaining auxiliary elements and main verb of the matrix clause, as defined by (65).

(64) C → M| HAVE| BE1| BE2| DO

(65) a. CB → C IP
attribution:
C.*AGR* ← IP.AGR_m
C.*x* ← '−', for *x* = *mood, part, perf, pass*
IP.*x* ← C.x_m, for *x* = *AGR, mood, part, perf, pass, tense*
IP.*inv* ← C.*inv*

b. C → X, for X = M, HAVE, BE1, BE2, and DO
attribution:
C.*inv* ← +
X.*x* ← C.*x* , for *x* = *AGR, mood, part, perf, pass, tense*
C.x_m ← X.x_m , for *x* = *AGR, mood, part, perf, pass, tense*
condition:
C.*root* = +
C.$tense_m$ = −

c. C → ϵ| for| that| whether
attribution:
C.*inv* ← −
C.AGR_m ← if C.*tense* = + then C.*AGR* else *nil*
C.x_m ← C.*x* , for *x* = *aux, mood, part, perf, pass, tense*

The category C has inherited attributes *AGR*, *mood*, *part*, *perf*, *pass*, *tense* and synthesized counterparts *inv*, AGR_m, $mood_m$, $part_m$, $perf_m$, $pass_m$, and $tense_m$. By the attribution (65.a), the inherited attribute occurrences *mood*, *part*, *perf*, *pass*, *tense* are negatively specified. If Subject-auxiliary inversion does not take place, a production in (64.c) applies and the inherited attributes of C are copied to the corresponding synthesized occurrences at C and get transferred

---

30 Subject-Aux inversion is really a misnomer in the present account, since we use no transformation.

to the IP node. By the attribution in (63), the attribute values are further transferred to the IB and AUX nodes. Hence, the first verbal element in the sentence must be [$-part$, $-perf$, $-pass$], and it is tensed or not, according to whether the clause it heads is tensed. We obtain the grammaticality assignments in (66).

(66) a. I may leave / I have left / I am leaving / I do leave

b. * I been leaving / * I done leave

Suppose now an auxiliary item is inserted under C, by application of (65.b) -- i.e., Subject-auxiliary inversion has taken place. By the first attribute condition in (65.b), is possible only in the matrix clause, for which C.*root* is positive. By the attribution rules there, the auxiliary category under C must be [$-part$, $-perf$, $-pass$] and the lexical item inserted under the subcategory must bear the same feature values. The insertion condition in (61) ensures that the auxiliary element under C behaves like the first auxiliary element in the Chomsky-like expansion (58). This predicts the grammaticality assignments in (67), which parallel those in (66).

(67) a. May I leave / Have I left / Am I leaving / Do I leave

b. * Been I leaving / * Done I leave

The last attribution rule in (65.b) copies the synthesized attributes occurrences $mood_m$, $part_m$, $perf_m$, and $pass_m$ of the auxiliary subcategory to the corresponding occurrences at the C node. By (64.d), these attribute occurrences are copied into the inherited occurrences *mood*, *part*, *perf*, and *pass* of the IP node; by the attribution in (64.a-b) the values are transferred to the IB and AUX nodes. In this manner, the auxiliary item inserted under C is effectively related to the rest of the auxiliary system of the clause: The auxiliary in complementizer position creates, via the synthesized occurrences [$part_m$, $perf_m$, $pass_m$], a lexical insertion condition for the first auxiliary element in IP. For example, if *have* is inserted under C, the next element must be [$+part$, $+perf$, $-pass$]. We obtain the grammaticality assignments in (68).

(68) a. Have you been seen / Have you sang

b. * Have you being seen / * Have you singing

Morphological feature propagation from the last auxiliary element in the clause (either in complementizer or inflection position), if any, to the main verb position is achieved by the attribution (60) and (63.b). In particular, by (63.b) the attribute occurrences [*part, perf, pass*] of the AUX node are inherited by VP, and ultimately by the head V. Hence, the morphology of the main verb must satisfy the requirements of the last auxiliary, if any. We obtain the examples (69).

(69) a. I may leave / I have left / I am leaving / I do leave

b. * I may leaving / * I have leave / * I do leaving

It remains to see how the linear order of auxiliary elements in a sentence is determined. If no Subject-auxiliary inversion takes place, the productions in (58) capture the observed order, as in the traditional analysis. If Subject-auxiliary inversion is present, we must consider the effect of the auxiliary in complementizer position. In this case, a production from the schema (65.b) must apply and $C.inv_m = +$. The attribution (60) for expansion of the AUX symbol has several attribute conditions, as in (70), which select the expansion of AUX that should be used, according to the material under C. If no auxiliary is under C, any of the three productions in (70) may apply. If an auxiliary is present, we assume a "trace" in AUX of the auxiliary in complementizer position and either (70.a) or (70.b) applies, according to whether the category under C is from the right-hand side in (70.a) or (70.b). The conditions account for the ungrammaticality in (71).

(70) a. AUX → M HAVE BE1 BE2
condition:
$AUX.inv = '-' \vee (AUX.mood = '+' \vee AUX.part = '+')$

b. AUX → DO
condition:
$AUX.inv = '-' \vee (AUX.mood = '-' \wedge AUX.part = '-')$

c. AUX → ϵ
condition:
$AUX.inv = '-'$

(71) * Do you may go / * Do you have gone / * Have you do go/gone

The lexical insertion rule (61) is also involved in the account of linear order. In Chomsky's (1957) account (51) of the auxiliary, the tense morphemes are base-generated before any other verbal element in a clause. Only one may be generated in a given clause, and it affix-hops to the first verbal element. This detail is incorporated in the last attribution rule of (61.a), which suppresses the *tense* feature (sets it to $-$) whenever a verbal element is inserted. In particular, an auxiliary in C position suppresses the *tense* feature for the remaining verbal elements in the matrix clause. Assuming that every modal element is tensed, we obtain that their occurrence is possible only in first verbal position in a clause, possibly the one in complementizer position. This explanation is identical to that of Gazdar, Pullum, and Sag (1982). Similarly, if we assume that auxiliaries of category HAVE are [$-$*part*, $-$*perf*, $-$*pass*], it follows that an item under HAVE may follow a modal, but not an auxiliary under any other auxiliary category.[31] As shown in table (62), the categories HAVE, BE1, and BE2 require [$+$*part*] or [$+$*perf*] successors. We obtain the grammaticality assignments in (72).

(72) Could he have done it / * Has he had done it / * Was he having done it

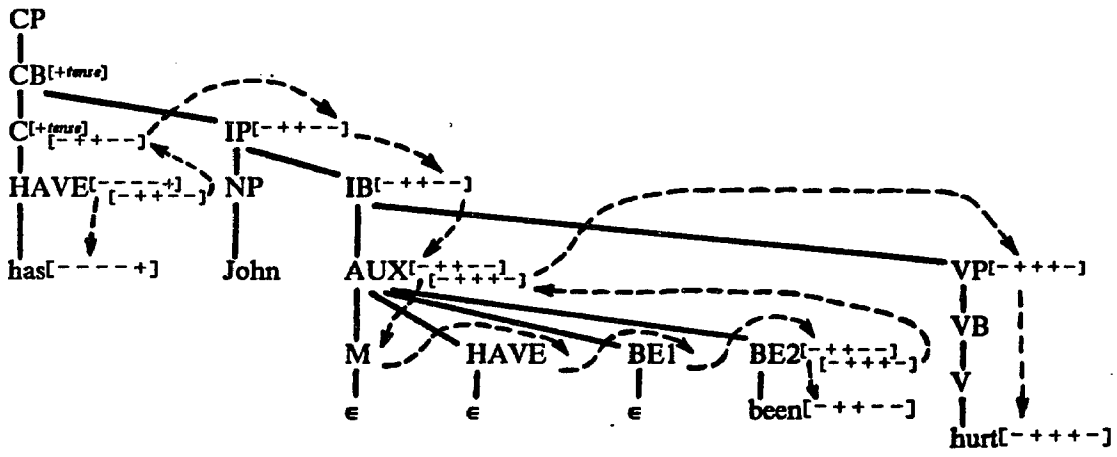
We will ignore here two minor attribute conditions which ensure the correct order of auxiliaries of category BE1 or BE2, relative to auxiliaries in C position.

In (73) we show two syntactic trees that display the principal auxiliary attributes associated with tree nodes, and their values. Example (73.a) is of a sentence with subject-auxiliary inversion, while (73.b) shows the same sentence, but without inversion. The annotations superscripted to the nodes are 5-tuples [*mood*, *part*, *perf*, *pass*, *tense*], consisting of inherited attributes only; similarly, the annotations subscripted are the corresponding synthesized attribute bundle [$mood_m$, $part_m$, $perf_m$, $pass_m$, $tense_m$]. The values given next to the lexical items *has*, *been*, and *hurt* are the lexically specified attributes [*mood*, *part*, *perf*, *pass*, *tense*].

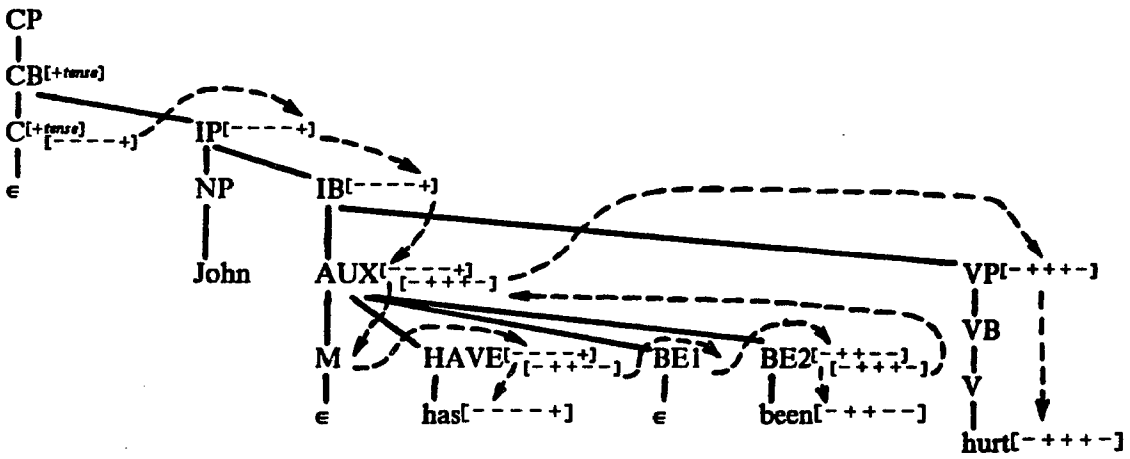
---

[31] Note that *have* may also occurr as a main verb, in which case the condition on the *part*, *perf* and *pass* features does not apply.

(73) a. Has John been hurt ?



b. John has been hurt.



3.5.4 Subject-verb agreement

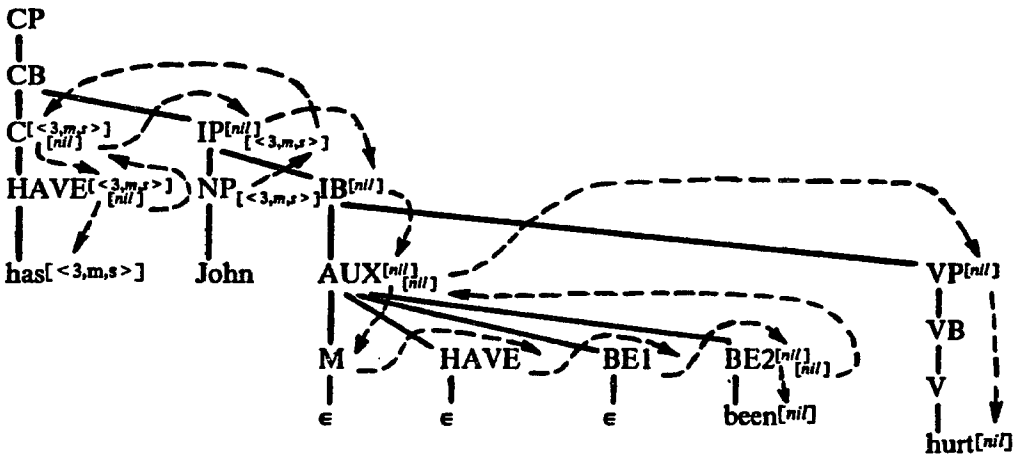
The attribution in (60), (61), (63), and (65) describe the process of Subject-verb agreement. The problem concerns the distribution of the agreement values *AGR* and AGR_m at node occurrences in a tree. The domain of the two attributes is triples of the form *<Person, Gender, Number>*, where *Person* is one of the numbers 1, 2, or 3, *Gender* is either *Masc*, *Fem*, or *Neuter*, and *Number* is either *plural* or *sing*. The special value *nil* is also in the domain of the agreement attributes.

AGR is inherited by the verbal categories, and in this case its value is determined by the context in which the node bearing the attribute appears. The attribute is also inherent in lexical items, and synthesized by NP, in which case its value is determined by the head noun. AGR_m, on the

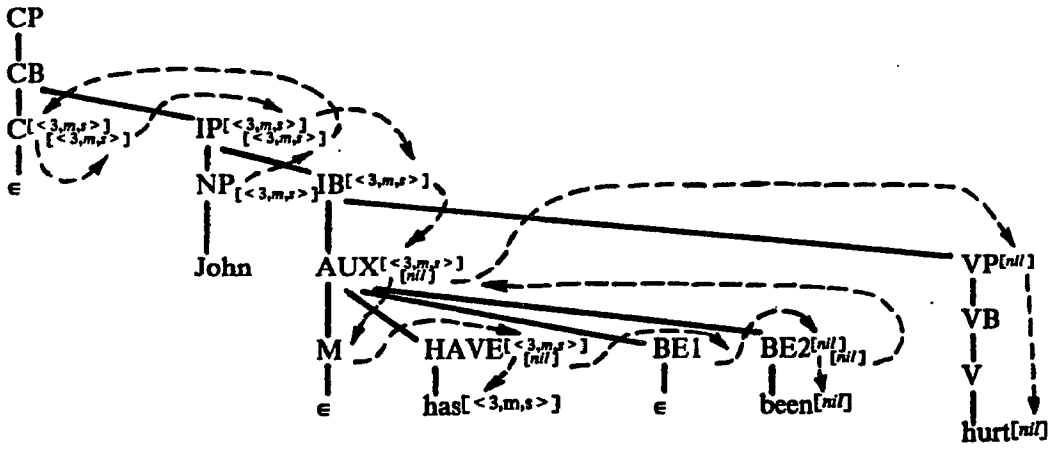
other hand, is associated with the verbal categories and synthesized; its value is determined by the lexical material dominated by the category, if there is any. The lexical insertion condition (61) is partly responsible for Subject-verb agreement. Insertion of an auxiliary form x under a node X is possible only if $x.AGR = X.AGR$.

Subject-verb agreement is obtained in general as follows. The basic process is transfer of the agreement features of the Subject to complementizer position. This is accomplished by the attribution (63.a) and (65.a). Thus, an auxiliary element appearing under C, if any, must agree with the Subject. Note that this element must be tensed, since inversion is possible only in the matrix clause, which must independently be tensed. If no auxiliary appears in C position and the clause is tensed (65.c), the inherited agreement feature of complementizer position is synthesized by the same position and propagated to the sentence, auxiliary, and verbal nodes, IP, IB, AUX, and VP, by attribution in (65.a), and (63). Hence, in this case the first verbal element in the sentence must agree with the Subject. Two illustrative syntactic trees are shown in (74), for a clause with Subject-auxiliary inversion (74.a), and for one without (74.b). The annotations superscripted to the nodes are the values of the inherited attributes AGR; the subscripts are the synthesized values of AGR_m.

(74) a. Has John been hurt ?



b. John has been hurt.



Note that after the first auxiliary element is found, or if the clause is untensed, the agreement value propagated through the remaining of the tree is *nil*; then any other verbal elements in the sentence bear no (non-*nil*) agreement. This is implemented by the attribution in (61.a) and (65.c), which cancels the agreement attribute after the first verbal element (61.a), or if the clause is untensed. Also, note that the transfer of agreement features from Subject to complementizer position is a local process. If (head-) movement of verbal elements turns out to be possible over longer distances in other languages, a more general mechanism will be necessary to relate the phrases that must agree. In cases where the Subject has been moved, we assume that the Subject trace has access to the agreement features of the antecedent defined by the Chain rule.

3.6 Logical Form and Binding

In Chapter 2 we had little occasion to discuss Logical Form; in this section we extend the remarks there and present a procedural model of Binding. Logical Form (LF) is a syntactic level of representation assumed in Government-binding, at which several aspects of the meaning of utterances is represented. LF representations are uninterpreted logical formulas in an, as yet, unspecified calculus. The rules that apply for the derivation of LF and the conditions on representations at that level define only the syntax of formulas in the LF calculus.

3.6.1 Logical Form

Logical Form is derived from S-Structure by two rules of interpretation: coreference assignment and operator scope assignment. The conditions on coreference assignment are defined by the binding theory. The rule for operator *scope assignment* is the rule Quantifier raising of May (1985). The denotations of words at LF are unanalyzed semantic monads. Only certain syntactic features of words are relevant at this level. The aspects of meaning represented at LF include those in (75):

(75) a. Predicate-Argument structure

b. Identification of sentential operators

c. Definition of operator scope

d. Conditions on coreference relations among overt nominals

Predicate-argument structure is defined by a complex process involving the lexicon, thematic role assignment, and move-α. Each component process is simple, but their interaction can give rise to quite complicated structures. Operators may be overt or not. Overt operators are lexical items marked by lexical features such as [+Q] (for Quantifier) or [+*Wh*] (for Wh-elements). All operators appear in C specifier or IP adjunct position at LF. Wh-operators may move to C specifier position in the mapping form D- to S-Structure, by application of move-α, as in Wh-question or relative clause formation. Remaining operators move to some IP adjunct position in the mapping from S-Structure to LF, by application of Quantifier raising. The scope of an operator is defined at LF by its c-command domain. Move-α and Quantifier raising coindex moved phrases with their traces; the coindexing defines operator-variable bindings. Lastly, coreference conditions among overt nominals are stated by the Binding theory, as we saw in Chapter 2.

Because of the nature of the items in (75), it is often said that Logical Form is a representation of the "structural meaning" of sentences (van Riemsdijk and Williams, 1986). The reader is referred to Chomsky (1977) and van Riemsdijk and Williams (1986) for more detailed discussion on the empirical nature and justification of Logical Form.

3.6.2 Binding

Binding theory assigns coreference to noun phrases by means of an indexing rule. The rule applies to an S-Structure and annotates it, assigning to every NP node in it a "referential index" that represents the coreference relation to the NP with other nominals in the sentence. Conditions on the output of the rule (binding axioms) apply at LF. It is still a research issue what the proper representation is for coreference relations -- i.e., what the "index" assigned is (cf. Chomsky, 1980; Chomsky, 1981; Higginbotham, 1986). As in Chapter 2, we adopt a notion of indexing in which two NPs are coreferential if they bear the same index. This is roughly the theory of Chomsky (1981).

The operation of the indexing rule is defined axiomatically by the Binding theory. Chomsky (1981) assumes a grammatical model for the derivation of LF in which the indexing rule applies freely, assigning randomly generated indices to noun phrases. Certain well-formedness conditions apply to LF representations, sanctioning, in fact, that certain index assignments are valid, while others are not. The well-formedness conditions fall under the subtheories of Binding and Control. These two subtheories are currently under intense study. A procedural refinement of the core ideas in the Binding theory is our concern in the sub-sections below.

3.6.3 Procedural Model of Binding

In this section we define a *Binding rule* that assigns to every NP node in an S-Structure a referential index, in such a way that the binding conditions are satisfied by the assignment. Those S-Structures for which there is no possible correct assignment are ill-formed, due to some violation of the binding theory.

3.6.3.1 The Binding rule

The new Binding rule is defined by attribution rules associated with productions in the base. The rule assigns indices according to the principles of the binding theory, and in this sense computationally implements it. Whenever the rule assigns a referential index, that index is guaranteed to satisfy the Binding principles. The new specification is more computationally efficient and psychologically plausible than the generate-and-test axiomatic method in Chomsky (1981). We believe the latter should be seen as a specification of some procedural counterpart, perhaps the Binding rule we define. The Binding rule applies to an NP after the functional type

of the NP has been determined, according to lexical features of its head.[32] As in Chapter 2, functional classification of an NP consists of evaluating the attributes *anaphoric* and *pronominal* of the NP.

The rule is non-deterministic, since it may derive one of several valid index assignments. This simply reflects the fact that several distinct (and valid) LF derivations are possible given the same starting S-Structure. We limit the scope of the indexing rule to those cases involving backward reference of anaphors and pronominals. Assignment of forward reference, as in (76), is not be covered by the rule. Also, the rule is such that it always assigns independent reference to referential expressions, ignoring cases where coreference with a previous phrase is possible. Extension of the rule seems feasible to accommodate the cases omitted; see (Correa, 1988).

(76) Every man who loved her$_i$ knew how kind Mary$_i$ was.

The indexing rule relies crucially on the following hypothesis: For every NP node in an S-Structure, it is possible to define two sets of nominal expressions consisting of, respectively, potential anaphoric antecedents, and potential pronominal antecedents. Each element in the sets is reference to a nominal that occurs elsewhere in the S-Structure. By the simplifying assumption that only backward reference is handled by the rule, each of the expressions is, furthermore, to the left of the NP at which the sets are defined (assuming a preorder enumeration of the nodes in the S-Structure).

Assuming a mechanism to compute the two sets noted, it is a straightforward task to complete the specification of the Binding rule. Conditions A, B, and C of the binding theory reduce to proper maintenace of the attributes representing the anaphoric and pronominal sets at the NP nodes -- conversely, we may regard the conditions as consequences of the manner in which these attributes are computed and propagated in a derivation tree, in the course of derivation of Logical Form. The Binding rule inspects the functional type of the NP node at which the rule applies. Depending on the values of the attributes *anaphoric* and *pronominal* (which define the type of the NP), the rule selects from the appropriate set of antecedents an element that agrees with the current NP. It is this step of the Binding rule which is non-deterministic. The

---

[32] The Binding rule applies only to overt nominals; these may become A-bound. Binding of non-pronominal empty NPs (traces) with their antecedents is defined by the Chain rule of section 3.2.

referential index of the element selected is then assigned to the current NP. Assuming an attribute *RefIndex* (referential index of a nominal), and attributes *AAS* (Anaphoric Antecedent Stack) and *PAS* (Pronominal Antecedent Stack) that represent the sets of potential anaphoric and pronominal antecedents of the NP, the indexing rule may be formulated as in (77).[33]

(77) **Binding Rule**

NP → ...

NP.*RefIndex* ← if NP.*anaphoric* = + then
if NP.*pronominal* = + then /* Control theory */
else *select-from*(NP.*AAS*)
else if NP.*pronominal* = + then *select-from*(NP.*PAS*)
else NP.*node*

The function *select-from* in (77) is non-deterministic; it takes a set (list) of expressions as argument and delivers the first element in the set whose agreement features match those of the current node. Hence, for pure anaphors (+*anaphoric*, -*pronominal*), the indexing rule selects an antecedent from the anaphoric antecedent stack (*AAS*), while for pure pronominals (-*anaphoric*, +*pronominal*), the rule selects an antecedent from the pronominal antecedent stack (*PAS*). If the expression is referential (-*anaphoric*, -*pronominal*), the rule assigns to the expression's its own root *node* number. By convention, when the referential index of the expression equals its node number the nominal is free in every domain: its reference is determined solely from the expression dominated by the node. If the NP is PRO, the antecedent, if any, is not determined by the indexing rule, but by the theory of Control.

We now proceed to describe in detail the types of the attributes involved in the computation, and the manner in which the *AAS* and *PAS* values are determined.

---

[33] The attribution (77) defines *RefIndex* as a synthesized attribute. The attribute is data-dependent only on other attributes of NP, so it may be defined with equal ease as an inherited attribute.

3.6.3.2 Binding attributes and their types

The attribute *RefIndex* is a positive integer which represents the referential index of the NP with which the attribute is associated. This attribute is synthesized, and its value is equal to the referential index of the leftmost NP with which the current NP corefers.

The attribute *AAS* (Anaphoric Antecedent Stack) records, for a given node in a tree, the sequence of c-commanding NPs found within the local domain of the node. Thus, any NP in this stack is a potential antecedent for the current node, if that node is anaphoric. Each element of the *AAS* is an ordered pair $<$NP.*RefIndex*, NP.$AGR_m$$>$, for some NP to the left of the current node. The nominals in the *AAS* are ordered in such way that the most recently found NP is on top of the stack. The order is defined by the tree traversal used for evaluation of the two attributes.[34]

The attribute *PAS* is similar to the *AAS*, except that each NP element in it either does not c-command the current node, or is outside the local domain of the current node. Each NP in the *PAS* is a potential antecedent for the current node, if that node is pronominal. An important difference between the *PAS* and *AAS* is that if the current node is a noun phrase, say NP_i, then the pair $<NP_i.node, NP_i.AGR_m>$ is a member of the *PAS*, but not of the *AAS*. Because of this, pronominals can be interpreted deictically, while anaphors cannot. If we assume that the c-command relation is irreflexive, then there is no need to stipulate as a special case this difference between the *AAS* and *PAS*.

We note that a third stack, say *RAS* (for Referential Antecedent Stack), of NPs not c-commanding the current node should be added to account for the anaphoric possibilities of referential expressions. However, we omit this detail in the formulation of the current Binding rule. As noted in the previous section, we always let the rule assign independent reference to referential expressions.

---

[34] No theoretical significance is attached to the order of the elements in the *AAS* or *PAS*. However, this order has practical consequences for the results delivered by the indexing rule (77). The ordering of the stack is indeed a ranking of the elements in it, from most to least "preferred antecedent" for the current node. Some psychological factors and other properties of sentences may contribute to this ordering. For example, psycholinguistic evidence suggests that gaps "reactivate" their antecedents.

Let us now illustrate by means of an example the distribution of values for the *AAS* and *PAS* attributes in an S-Structure. Consider the expression (78), in which the subscripts are noun phrase *node* numbers. For simplicity we assume that the NP nodes have been numbered *h*, *i*, *j*, and *k*, ignoring the actual value of the *node* attributes.

(78) $John_h$ told [his_i parents]$_j$ about $himself_k$.

(79)

a.	$NP_h.AAS = [\,]$	$NP_h.PAS = [<h,AGR_h>]$
b.	$NP_i.AAS = [\,]$	$NP_i.PAS = [<i,AGR_i>, <h,AGR_h>]$
c.	$NP_j.AAS = [<h,AGR_h>]$	$NP_j.PAS = [<j,AGR_j>]$
d.	$NP_k.AAS = [<j,AGR_j>, <h,AGR_h>]$	$NP_k.PAS = [<k,AGR_k>, <i,AGR_i>]$

The values that result for the *AAS* and *PAS* are as shown in (79). For the first phrase NP_h there is no potential anaphoric antecedent, so $NP_h.AAS$ is empty (denoted [ ]). $NP_h.PAS$ contains the node's own *node* number and agreement features $<h, AGR_h>$, as unique element, so at that position it is possible to have a free pronoun. This is shown with a parallel sentence in (80.a). For the second noun phrase, NP_i, the *AAS* and *PAS* values are as shown in (79.b). Thus, no anaphor is permissible at the position, since the *AAS* is empty, but a pronoun is, in which case it may be interpreted deictically (80.b) or anaphorically, refering back to NP_h (80.c). The *AAS* and *PAS* associated with NP_j are shown in (79.c). Thus an anaphor is possible if it can refer to NP_h (80.d); a pronominal may be used, if it is free (80.e). For the last NP, NP_k, the values are as shown in (79.d). There are two potential anaphoric antecedents, NP_h and NP_j, and both possibilities may be productively exploited as in (80.f-g). A pronoun at this position is also permitted, and may be free, or interpreted anaphorically, if it refers back to NP_i (80.h).

(80) a. *He*$_h$ told [his parents] about himself.

b. *John*$_h$ told [*his*$_i$ parents] about himself.

c. *John*$_h$ told [*his*$_h$ parents] about himself.

d. *John*$_h$ told *himself*$_h$ to stop smoking.

e. *John*$_h$ told *him*$_j$ to stop smoking.

f. *John*$_h$ told [his parents] about *himself*$_h$.

g. John told [*his parents*]$_j$ about *each other*$_j$.

h. John told [*Mary*$_i$'s parents] about *her*$_i$.

In addition to the attributes *AAS* and *PAS*, and in order to simplify the attribution rules that compute the value of *PAS*, it is convenient to posit a third attribute PAS_S, which is defined for every node that has the *PAS* attribute associated with it. In contrast with *PAS*, which is inherited, PAS_S is synthesized. The type of PAS_S is the same as that of *PAS*. Its value at a node X is derived from that of *PAS*, by two operations. First, we push into the new set the identifiers of all NPs embedded in the phrase dominated by X. It should be clear, by principles of X-bar theory, and since NP is a maximal projection, that for any X, if NP is dominated by X, then there is a branching node on the path connecting NP and X. Thus the embedded NP does not c-command any node c-commanded by X. It follows that the new NPs pushed into PAS_S are potential antecedents for pronominals in the c-command domain of X, even if X is within the same local domain as the pronominal and c-commands it.

We illustrate the first operation on PAS_S with the S-Structure (81). This structure is identical to (78), except that it has a matrix Subject with an embedded nominal, namely *a donkey*. The values of the *PAS* and PAS_S attributes for the first two NP nodes are as shown in (82). NP_g is embedded within NP_h, and so a reference to it may be synthesized in $NP_h.PAS_S$ (82.b), and made available as a potential pronominal antecedent to NP_k. It is therefore possible for NP_k to corefer with NP_g.[35] The attribute PAS_S is also involved in the computation that yields the values of the attribute *PAS* in (79). For example, NP_i is available as a potential pronominal antecedent of NP_k because NP_i is embedded in NP_j, and so may be synthesized by PAS_S.

(81) [Every man who owned [a donkey]$_g$]$_h$ told [*his*$_i$ parents]$_j$ about *it*$_k$.

(82)

a. $NP_g.PAS = [<g,AGR_g>]$ $\quad$ $NP_g.PAS_S = [<g,AGR_g>]$

b. $NP_h.PAS = [<h,AGR_h>]$ $\quad$ $NP_h.PAS_S = [<g,AGR_g>, <h,AGR_h>]$

The second operation done to evaluate a PAS_S attribute occurrence at a node X is to remove from the original *PAS* any element that has been coindexed with the current node X. In (83), the NP *John* is a potential antecedent for either of the pronouns *he* or *him*. However, in (83.a)

---

[35] Another interpretation of (81) lets the pronominal NP_k, *it*, refer to the clause *owned a donkey*, rather than to *a donkey*. This is CP anaphora, about which current Government-binding theory does not say much.

he has been coindexed with *John*, so the later is not anymore available as an antecedent for *him*. In (83.b), where *he* is interpreted deictically, *him* may be coindexed with *John*. The result in (83.c) is ungrammatical, since both pronouns are coindexed with *John*. Technically, the binding of *John* and *him* in (83.c) satisfies condition B of the binding theory, but the binding of the two pronouns *he* and *him* is local, so it does not. The data in (83) is explained if we assume, as above, that an element in the *PAS* becomes unavailable (within a local domain) as a potential pronominal antecedent if it becomes coindexed with a c-commanding node (within the current local domain). Recall that the indexing rule (77) is non-deterministic, so it is impossible to tell which *PAS* element should be removed before the indexing rule applies. The value of the attribute PAS_S of an NP is data-dependent on the *RefIndex* attribute of the same NP.

(83) a. $John_i$ said that he_i would kill him_j

b. $John_i$ said that he_j would kill him_i

c. * $John_i$ said that he_i would kill him_i

3.6.3.3 Attribution rules

Having described the attributes *AAS*, *PAS*, and PAS_S of the procedural Binding rule, it seems best to proceed to give the attribution rules that define how their values are to be computed.

Rules for CP ***and*** IP***:*** In the start production (84.a), the attribution rules indicate that both *AAS* and *PAS* are empty at the root node. This condition simply indicates that there are no potential antecedents for anaphors or pronominals outside the matrix clause. Government-binding grammar is a sentence-level grammar; it does not deal with larger discourse entities, and so we do not consider possible intersentential relations.[36] If the clause is untensed, the *AAS* and *PAS* attributes percolate downwards, by the attribution associated with the productions that expand CP (84.b) and CB (84.c). Meanwhile, the synthesized attribute PAS_S is propagated upwards by the same rules. If the clause is tensed, an interesting computation takes place in the CP production (84.b). First, *AAS* is reset to empty. Thus anaphors within a tensed clause cannot

---

[36] In a grammar that deals with larger discourse entities than sentences, the root *PAS* occurrence may evaluate to a non-empty value, making preceeding phrases available as potential antecedents for pronominals in the current clause.

find antecedents outside it, as required by binding condition A. On the other hand, the value for *PAS* at the CB node is computed by union of the values of the old *AAS* and *PAS* at the CP node. This is indicated by the union operator '⋃' in (84.b). Thus, pronominals within the current clause may take antecedents which were previously available only to anaphors; this situation is illustrated by the contrast between (85.a) and (85.b). Notice that the optional NP specifier in (84.b), which is the landing site for Wh-movement, does not enter into the computations for *AAS* and *PAS*. A phrase in A-Bar position is not a potential antecedent.

(84) a. $Z \rightarrow CP$

attribution:

$CP.AAS \leftarrow [\,]$

$CP.PAS \leftarrow [\,]$

b. $CP \rightarrow (NP)\ CB$

attribution:

$CB.AAS \leftarrow$ if $CP.tense = +$ then $[\,]$ else $CP.AAS$

$CB.PAS \leftarrow$ if $CP.tense = +$ then $CP.AAS \cup CP.PAS$ else $CP.PAS$

$CP.PAS_S \leftarrow CB.PAS_S - CP.AAS$

c. $CB \rightarrow C\ IP$

attribution:

$IP.AAS \leftarrow CB.AAS$

$IP.PAS \leftarrow CB.PAS$

$CB.PAS_S \leftarrow IP.PAS_S$

d. $IP \rightarrow NP\ IB$

attribution:

$NP.AAS \leftarrow IP.AAS$

$NP.PAS \leftarrow [<NP.node, NP.AGR_m> \mid IP.PAS]$

$IB.AAS \leftarrow [<NP.RefIndex, NP.AGR_m>]$

$IB.PAS \leftarrow NP.PAS_S \cup IP.AAS$

$IP.PAS_S \leftarrow [<NP.RefIndex, NP.AGR_m> \mid IB.PAS_S]$

(85) a. * *John*$_i$ expects [*him*$_i$ to win]

b. *John*$_i$ expects [that *he*$_i$ will win]

Another interesting set of attribution rules is that associated with the IP production (84.d). The *AAS* of the Subject (NP) is inherited from the IP node. If this antecedent stack is empty (e.g., due to a tensed clause), an anaphor is impossible in Subject position. If there is no potential antecedent, the indexing rule (77) would signal an illicit configuration, namely, one in which it is asked to retrieve an antecedent from an empty stack. The *PAS* of the Subject, on the other hand, is formed from the *PAS* of the IP node, by pushing into it the Subject's own *node* number and agreement features $<$NP.*node*, NP.$AGR_m$$>$. The notation [$x$| Y] is used to represent a stack of elements whose top element is x and whose tail is Y. Notice that x is an element, while Y is a stack of elements, namely that obtained from [x| Y] by removing the first element. A pronominal Subject is therefore always possible, since the *PAS* is never empty. Where the referential index of the Subject is set to NP.*node*, we say that it is referentially independent.

Moving to the IB node, its *AAS* is formed from the *AAS* of IP by pushing into it the identifier for the Subject. Hence the Subject becomes a potential antecedent for an anaphor in object position, as the VP attribution rules below will show. Notice that we use the Subject's referential index, rather than the *node* number, since the two may be different. The *PAS* of the IB node is simply copied from the synthesized PAS_S of the Subject. The PAS_S of the IP node, on the other hand, is derived from the PAS_S of IB, by pushing into it the identifier for the Subject node. Thus, the NP Subject becomes a (non c-commanding) potential antecedent for a pronoun to the right, as in (86).

(86) [The woman [who *Bill*$_i$ kissed]] said *he*$_i$ was terrific.

Rules for* V *and its Projections: The attribution rules for the projections of V, given in (87), are more regular than the first set (84) of rules. The *AAS* and *PAS* attributes are percolated downwards form the IB to the VB node. Meanwhile, the PAS_S is passed upwards from VB to IB. This is specified by attribution rules (87.a-b). Three of the rules for VB are (87.c-e). For verbs that take no complements, the value of the synthesized attribute PAS_S is *PAS*. The attributes *AAS* and *PAS* remain otherwise unused. For transitive verbs and verbs that take a clausal complement, the *AAS* and *PAS* are transfered from the VB node to the root node of the

complement. The expressions in the two stacks become potential antecedents for an NP complement (88.a), or for nominals embedded in a clausal complement (88.b).

(87) a. IB → I VP

attribution:

VP.*AAS* ← IB.*AAS*

VP.*PAS* ← IB.*PAS*

IB.PAS_S ← VP.PAS_S

b. VP → ... VB ...

attribution:

VB.*AAS* ← VP.*AAS*

VB.*PAS* ← VP.*PAS*

VP.PAS_S ← VB.PAS_S

c. VB → V

attribution:

VB.PAS_S ← VB.*PAS*

d. VB → V XP, for XP = NP, CP

attribution:

XP.*AAS* ← VB.*AAS*

XP.*PAS* ← VB.*PAS*

VB.PAS_S ← if XP = NP then [< NP.*RefIndex*, NP.AGR_m >] else XP.PAS_S

e. VB → V NP XP, for XP = PP, CP

attribution:

NP.*AAS* ← VB.*AAS*

NP.*PAS* ← VB.*PAS*

XP.*AAS* ← [< NP.*RefIndex*, NP.AGR_m > | VB.*AAS*]

XP.*PAS* ← NP.PAS_S

VB.PAS_S ← XP.PAS_S

(88) a. [*John*$_i$'s teacher]$_j$ hurt *himself*$_j$/*him*$_i$.

b. [*John*$_i$'s teacher]$_j$ believes [*himself*$_j$/*him*$_i$ to be a fine man]

If the complement in (87.d) is NP, that NP becomes part of the synthesized PAS_S of the VB node, by pushing its identifier into NP.PAS_S. In this manner an NP in object position may be an antecedent for a pronominal to its right (89.a). Otherwise the PAS_S is identical to the PAS_S of the complement. Finally, for verbs that take both an NP and a PP or clausal complement (87.e), the rules operate as before, except that the first complement NP becomes a potential antecedent for anaphors or pronominals within the second complement (89.b). As with the rules (87.d), the NP complement is a potential antecedent for pronominals that follow to the right, so its identifier is pushed into the PAS_S of the VB node.

(89) a. [The woman [who kissed *Bill*$_i$]] said *he*$_i$ was terrific.

b. Bill told *Mary*$_i$ [that *she*$_i$ is pretty]

Rules for N ***and its projections:*** We now turn our attention to the attribution for NP and its projections, starting with the PAS_S attribute. As may be inferred from the discussion above, there are four possible cases regarding the value of PAS_S, at some node NP$_i$. If NP$_i$ contains other embedded nominals, then the value of PAS_S will contain identifiers (referential index and agreement features) for the embedded nominals (cf. (81) above). If NP$_i$ is a bound pronominal, refering to some previous NP, say NP$_j$, which must be in NP$_i$.*PAS*, then PAS_S will have NP$_j$ removed from it (cf. (83)). These two cases are independent of each other, and thus we obtain four possibilities regarding the value of NP$_i$.PAS_S. The value of PAS_S is identical to that of NP$_i$.*PAS* if and only if NP$_i$ does not contain any embedded NPs and is referentially independent (NP$_i$.*RefIndex* = NP$_i$.*node*).

The attribution rules that specify this behavior, together with the rules that define the values of the *AAS* and PAS attributes for nodes internal to NP, are now given in (90). Rule (90.a) is for the case where the NP specifier is either absent or a determiner. Rule (90.b) is for noun phrases with NP specifier.

(90) a. NP → (Det) NB

attribution:

NB.AAS ← NP.AAS

NB.PAS ← tail(NP.PAS)

NP.PAS_S ← NB.PAS_S

b. NP_1 → NP_2 NB

attribution:

NP_2.AAS ← NP_1.AAS

NP_2.PAS ← [< NP_2.*node*, NP_2.AGR_m > | tail(NP_1.PAS)]

NB.AAS ← [< NP_2.*node*, NP_2.AGR_m >]

NB.PAS ← NP_2.PAS_S

NP_1.PAS_S ← [< NP_2.*RefIndex*, NP_2.AGR_m > | NB.PAS_S]

c. NB → N

attribution:

NB.PAS_S ← NB.PAS

d. NB → N XP, for XP = PP or CP

attribution:

XP.AAS ← NB.AAS

XP.PAS ← NB.PAS

NB.PAS_S ← XP.PAS_S

In production (90.a), where there is no nominal NP specifier, the AAS and PAS attributes are essentially percolated from the NP to the NB node. The attribute PAS is not strictly percolated, as its top is removed in the transfer. This is accomplished by the stack function *tail*. To see why the top of PAS must be removed as it is percolated from NP to NB, recall that the top is the NP identifier. This condition permits a deictic interpretation of NP, in case it is pronominal. However, phrases within the NP cannot refer to it, as shown by the ill-formedness of (91). Removal of the top element in PAS is part of a computational explanation of the *i*-within-*i* condition, which is traditionally used to rule-out (91). From the removal of the top of PAS at a node it follows also that the occurrence of PAS_S at the node does not contain the identifier of NP (see the attribution rules (90.b,c)).

(91) a. * [a book about it_i]$_i$

b. * [a book about *itself*$_i$]$_i$

The value of PAS_S in (90.a) is simply copied from the NB node, since there any embedded NP is dominated by NB also.

In contrast, in (90.b), where NP_1 has an NP specifier NP_2, the synthesized PAS_S is computed from the PAS_S on the NB symbol, by pushing into it the identifier for NP_2. Thus NP_2 becomes a potential antecedent for pronominals to the right of NP_1, as in (92.a). Notice that NP_2 does not c-command any constituent to the right, as required by binding condition B. The process may apply recursively, as in (92.b), where *him* may refer to either *John* or *John's child.*

(92) a. [*John*$_i$'s teacher] punished *him*$_i$

b. [[*John*$_i$'s *child*]$_j$'s teacher] punished *him*$_i$/$_j$

The inherited attributes *AAS* and *PAS* of NP_2 in (90.b) also offer interesting characteristics. Both attributes are essentially percolated from NP_1, the parent node. If the corresponding anaphoric and pronominal stacks are non-empty, it is possible to have an anaphor or a pronoun at the position, as in Chomsky's examples (96.a,c) discussed below. Both of those examples are problematic for the binding theory. It should be noted that if NP_2 is an anaphor, must be a reciprocal, not a reflexive (by some yet to be discovered condition; a constraint against genitively inflected reflexives would do). The *PAS* of NP_2 is not strictly percolated from NP_1. In (90.a), the top of NP_1.*PAS* is removed, and the NP identifier of NP_2 is pushed. Thus, a pronoun in NP_2 position cannot take NP_1 as antecedent (93.a) (an *i*-within-*i* violation), and it can be interpreted deictically (93.b). We leave it as an exercise to explore the empirical consequences of the value assignments to the *AAS* and *PAS* attributes of NB in (90.b). The PAS_S attribute is computed as usual, except that NP_2 is pushed into it, thus becoming available as a potential antecedent for pronominals to the right, much as the I specifier in (84.d).

(93) a. * [*His*$_i$ brother]$_i$

b. [*His*$_j$ brother]$_i$

The last two NP rules we consider are (90.c,d), both productions for NB. In (90.c) N has no complements and the synthesized PAS_S is equal to the inherited *PAS*. In (90.d) N has a complement, either PP (including *of*-NP phrases) or CP. The *AAS* and *PAS* attributes are copied as usual from the parent NB node to the complement, and in this manner anaphors (94.a) or pronominals (94.b) within the complement may find their antecedents. If the complement is a clause in (90.d), it must either be tensed or have a PRO Subject, since N is not a Case-assigner. The *AAS* attribute is actually not used, since in the first case it is reset to empty, according to attribution rules (84.b), and in the second the PRO controller is determined according to the principles of the Control theory. Finally, the PAS_S synthesized by NB, is obtained from the complement, and contains the identifier of all NPs within the complement.

(94) a. *The Nibelungs*$_i$ sang [stories [about *themselves*$_i$]]

b. *The Nibelungs*$_i$ fought for [the ring [Sigfried stole from *them*$_i$]]

Rules for* P *and its projections: The attribution rules responsible for propagating the *AAS* and *PAS* attributes in PP phrases appear in (95). We spare the detail of going through them; instead we assume familiarity with the attributes and mechanisms involved to understand the new rules without further explanation.

(95) a. PP → PB
attribution:
PB.*AAS* ← PP.*AAS*
PB.*PAS* ← PP.*PAS*
PP.PAS_S ← PB.PAS_S

b. PB → P NP
attribution:
NP.*AAS* ← PB.*AAS*
NP.*PAS* ← PB.*PAS*
PB.PAS_S ← [< NP.*RefIndex*, NP.AGR_m > | NP.PAS_S]

3.6.4 On the difference between Binding and movement

The problematic exceptions to the binding theory in Chapter 2, in (136), are not the only ones. In fact, there is at least one more possibility, as shown in (96). In all three cases, it is possible for the pronoun and the anaphors to be coindexed with *the children*, as antecedent, without any apparent violation or ungrammaticality.[37]

(96) a. *The children*$_i$ thought [$_{CP}$that [$_{NP}$ [each other$_i$]'s pictures] were on sale]

b. *The children*$_i$ thought [$_{CP}$that [$_{NP}$ pictures of [*each other*$_i$]] were on sale]

c. *The children*$_i$ thought [that [$_{NP}$ *their*$_i$ pictures] were on sale]

As with the exceptions in Chapter 2, a strict interpretation of the notion Minimal Governing Category in Chapter 2 yields the Subject NP of the embedded clause as the MGC for the pronoun in (96.c) and the anaphor in (96.a). Thus (96.c) satisfies the binding conditions, but (96.a) does not. For (96.b), the MGC of the embedded anaphor would be the IP node of the embedded clause. Even in this case, the binding condition for anaphors is violated, since the anaphor is bound outside its MGC.

Again, we do not try to resolve these difficulties in the binding theory; instead, we refer the reader to Chomsky (1986.a) for discussion of the problem. The examples (96.a-b) are of interest to us for another reason. They are instances of so-called "long-distance binding," and illustrate well the difference between anaphoric binding and movement. We would like to use this difference to justify the different implementations (or computational solutions) we developed for the movement and binding theories.

The question we have in mind is the following: Given that empty categories, and in particular NP-trace and Wh-trace, are classified by the features ±*anaphoric* and ±*pronominal*, as lexical NPs are, why is it not appropriate to derive antecedent-trace relations according to the principles of the binding theory, while dispensing of the theory of movement and the Chain rule that computationally describes it? In answer to this question, the first point that should be made is

---

[37] Examples (96.a-b) are Chomsky's (1986a); (113) in Ch.3.

that the theory of binding applies only to argument-binding -- i.e., those cases where the antecedent of the anaphor is in argument position. Hence, the binding theory does not account for the distribution of Wh-traces (variables) in syntactic trees, and does not contribute to the determination of Wh-trace-antecedent relations. For this reason alone an independent theory of movement is necessary.

Instances (96.a-b) of long distance binding provide another justification. While (96.b) is grammatical, the structurally parallel string (97.a), whose D-Structure is (97.b), is ungrammatical. Anaphoric binding of the two italicized positions in (96.b) is possible, but chain indexing due to movement is not. The ungrammaticality of (97.a) follows from the violation of two principles, namely subjacency, and the complex-noun-phrase constraint, to which A-binding is not subject.

(97) a. * *The children*$_i$ seem [that [$_{NP}$ pictures of [$\in$]$_i$] were on sale]

b. It seems [that [$_{NP}$pictures of [the children]] were on sale]
(Example (a) is Chomsky's (1986a) (116) of Ch.3.)

These observations provide support for independent theories of movement and binding, and also good justification for implementing the Chain rule ("movement") with a system of attributes and attribution rules unrelated to those used to implement the Binding rule. The two theories achieve nearly identical results regarding argument binding, but this is partly an accident, due to the cross-classification of lexical and empty categories by the features $\pm$*anaphoric* and $\pm$*pronominal*. One might predict that every instance of A-Chain indexing (due to movement) is compatible with the binding theory, but not viceversa, as (96.b) and (97.a) illustrate.

3.7 Summary and conclusion

In this chapter we presented a method for attribute grammar (AG) definition of the grammatical processes in Government-binding theory. This was done by implementing a significant fragment of the theory, as defined in the current literature. The relation between the original theory and the AG statement is clearly defined. However, the attribute grammar statement is incredibly detailed and includes many details of realization not present in the original abstract theory. The fact that the attribute grammar is incredibly detailed makes it easily falsifiable. This need not be

taken as a weakness of the AG statement. Instead, more precise and satisfactory statements of Government-binding may be used to enhance the AG definition.

The attribute grammar presented is a computational statement of GB theory. It assumes explicit sets of grammatical categories, attributes, and directed attribution rules for attribute evaluation. This is in sharp contrast to the specificational or *principle-based* approach of Government-binding, in which attribution rules are extremely trivial, of the sort *"assign any value in its domain to an attribute,"* while the bulk of the grammar is in attribute conditions that constraint the values of the attributes assigned. We contrast the generate-and-test approach of Barss' (1983) Chain Binding rule and Chomsky's (1981) Free-Indexing rule in Binding to the directed approach of the Chain and Binding rules developed above in the chapter. The principle-based approach is attractive for its simplicity and ability to ignore questions of implementation. The computational model, on the other hand, is attractive as model of performance and for the further insight it brings into grammar. Also, given our current models of computation, specifications are rarely useful as practical models of performance; substantial creative work is often needed to turn a given specification into a practical equivalent one.

The attribute grammar developed in this chapter differs in several respects from the Government-binding model of Chapter 2. This includes coverage, use of non-binary-branching rules, and omision of several principles of grammar, such as the ECP. The Binding rule also contains shortcomings in its handling of movement structures. Correa (1988) gives a revised version of the rule which solves some of these problems.

As noted above, the attribute grammar developed here is a computational model of natural language grammar. But it does not yet define an algorithm for natural language analysis. In the following chapter we develop one such algorithm, among the space of many possible.

Chapter 4. Parsing Government-binding Grammar

In Chapter 3 we developed an attribute-grammar for a substantial fragment of English, following the notions and principles of the Government-binding theory. To this formal statement of the theory a number of standard methods and tools may be applied to derive automatic syntactic analysis or *parsing* procedures. In the present chapter we undertake this task.

In section 1 we state basic definitions of context-free grammars and languages and several notions associated with them. We do not, of course, identify natural languages or the languages that may be defined by Government-binding grammars with the class of context-free languages. For us this is an open question, whose answer is likely to depend to large extent on idealizations about language and speakers. The definitions in section 1 are simply part of the mathematical notation that is convenient to discuss sets of strings. In section 2 we summarize the phrase-structure component of the attribute grammar, and point out some of its properties. In section 3 we examine pushdown automata and their use in models for parsing; the section ends with the formulation of our basic parsing algorithm. The algorithm we develop is top-down with one symbol lookahead and constructs left-most derivations first, scanning the input from left-to-right. In section 4 we summarize the attribute system of the grammar and define the attribute evaluation scheme. In section 5 some extensions of the basic parsing algorithm of section 3 are developed, in order to improve its runtime efficiency. In section 6 we discuss the performance of the extended parsing procedure, considering worst and typical cases, and present some sample runs.

The formal methods and results used in this chapter to arrive at a parsing algorithm have been developed by computer scientists and linguists over the past thirty years. Thus, the reader should not expect new results here. Harrison (1978), Hopcroft and Ullman (1979) and Waite and Goos (1984) present many of the mathematical results and formal methods we use.

An item of novelty and interest in this chapter, however, is the attribute evaluator we develop. It differs from previous ones (e.g., Kastens, Hutt, and Zimmermann, 1982; Madsen, 1980) in that it can be applied in small steps, in combination with the derivation steps provided by the underlying context-free grammar. This feature of the evaluator is important for the parser, since the underlying grammar is extremely ambiguous. Another aspect of interest here is the extensive application of previously established results and methods to the parsing problem for Government-binding grammars. No attempt has been made in the chapter to incorporate results from the psycholinguistics literature in order to arrive at the parsing model adopted. The choice of a top-down parsing algorithm should not be construed as endorsement of this particular model as a psychological model of how people actually parse strings in their language. The reader is refered to Kimball (1972) and Fodor (1978) for discussion of human parsing strategies.

4.1 Definitions

Let T be a finite vocabulary of *terminal symbols* or words. The *Kleene closure* of T is the set $T^* = \{t_1 t_2 ... t_k : k \geq 0, t_i \in T, \text{for } i = 1, ..., k\}$ of finite strings which may be formed by concatenating words from T. A *language* over T is a (finite or infinite) subset of T^*. If L is a language, each element $s \in L$ is called a *sentence* of L. If L is a finite language it can in principle be described by listing each of its sentences; this mode of description would however not be economical and fail to capture any regularities and generalizations about the language. If L is infinite, the former option is not open for obvious reason, and instead a finite grammatical device must be found to describe the language. Context-free grammars (CFGs) are formal devices for language description, as defined in (1).

(1) A *context-free grammar* is a quadruple G=(N, T, P, Z), where T is a finite set of *terminal symbols*, N a finite set of *nonterminal symbols* or syntactic categories, Z a distinguished symbol of N called the *start symbol*, and P a finite set of productions of the form $X \rightarrow \alpha$, where $X \in N$, and $\alpha \in (N \cup T)^*$. The set $V = N \cup T$ is called the *vocabulary* associated with G.

Each syntactic category of a CFG defines a language. The languages represented by the categories are defined recursively by the production set, in terms of each other and the terminal symbols of the grammar. Each production states that the language associated with the category on its left-hand side contains strings formed by concatenation of strings in the languages of

certain other categories, along possibly with some terminal symbols. The language represented by the start symbol is identified as the language generated by the grammar.

In order to define precisely the language generated by a grammar G we need a binary relation $\Rightarrow$ (2) which relates strings over the vocabulary of the grammar. We let $\Rightarrow^*$ denote the reflexive and transitive closure of $\Rightarrow$. These relations are crucial in the notion of derivation (3).

(2) Let $G=(N, T, P, Z)$ be a context-free grammar. We define two binary relations $\Rightarrow$ and $\Rightarrow^*$ over V^*. If $\alpha X\beta$ and $\alpha\gamma\beta$ are strings in V^* and $X\rightarrow\gamma$ is a production in P, then $\alpha X\beta \Rightarrow \alpha\gamma\beta$, and we say that $\alpha X\beta$ *directly derives* $\alpha\gamma\beta$. The relation $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$. Given strings $\alpha_1, \alpha_k \in V^*$, we say that $\alpha_1 \Rightarrow^* \alpha_k$, or α_1 *derives* α_k, if there exist strings $\alpha_2, ..., \alpha_{k-1}$ such that $\alpha_i \Rightarrow \alpha_{i+1}$, for $i=1, ..., k\text{-}1$.

(3) Given $G=(N, T, P, Z)$, a sequence $\alpha_0, ..., \alpha_k$ of strings in V^* is called a *derivation of length k* if $\alpha_0=Z$, and $\alpha_i \Rightarrow \alpha_{i+1}$, for $i=0, ..., k\text{-}1$. If at each step i in the derivation a production is applied to expand the left-most nonterminal in α_i, the derivation is said to be *left-most.* Similarly, if it is the right-most non-terminal which is expanded at each step, the derivation is *right-most.*

Given the elements of definitions (2) and (3), the language generated by a context-free grammar may be defined as in (4).

(4) Let $G=(N, T, P, Z)$ be a context-free grammar. The *language* generated by G is the set $L(G)=\{ \omega: \omega \in T^* \text{ and } Z \Rightarrow^* \omega\}$ of strings that may be derived from the start symbol Z.

From the definitions above, a context-free grammar is a finite system which may describe an infinite language. A set L of strings is said to be a *context-free language* if there exists a CFG G such that $L=L(G)$. It is always possible to obtain a second CFG $G' \neq G$ such that $L(G)=L(G')$. In this case G and G′ are said to be (weakly) *equivalent.* The two grammars are called *strongly equivalent* if they provide the same set of structural descriptions (derivation trees) for each sentence of the language they generate.

4.2 The Phrase-structure Component

The phrase-structure component in the attribute-grammar of the previous chapter is a context-free grammar whose formulation has been influenced by the analyses of Jackendoff (1977), and incorporates current assumptions about X-bar theory (Chomsky, 1986b).

4.2.1 Context-free base

The context-free grammar assumed for a subset of English is shown in (5).

(5)

CP → CB	IP → NP IB
CP → NP CB	IB → I VP
CP → PP CB	IB → I AdvP VP
CB → C IP	I → ϵ | (M)(HAVE)(*to*)(BE1)(BE2)(DO
C → ϵ | for| that| whether| I	

NP → NB	VP → VB
NP → NP NB	VP → VB AdvP
NP → ArtP NB	VP → VB PP
NP → QP NB	VP → VB PP PP
NP → ArtP QP NB	VP → VB CP
NP → AP NB	VB → V
NP → ArtP AP NB	VB → V AP
NP → QP AP NB	VB → V PP
NP → ArtP QP AP NB	VB → V PP PP
NP → ϵ	VB → V NP
NB → N	VB → V NP PP
NB → N PP	VB → V NP NP
NB → N CP	VB → V NP CP
	VB → V CP

PP → PB	AP → AB
PP → ϵ	AP → DegP AB
PB → P	AB → A
PB → P NP	AB → A PP
PB → P PP	AB → A CP

AdvP → AdvB	QP → QB
AdvP → DegP AdvB	QP → DegP QB
AdvB → Adv	QB → Q
ArtP → ArtB	DegP → DegB
ArtP → DegP ArtB	DegB → Deg
ArtB → Art	

The base grammar (5) does not include productions to insert lexical items under lexical categories.[1] In the formulation of the parsing algorithm below, we let the terminal alphabet of the grammar consist of the *lexical* categories (6.a) in the grammar and the small set of *specified lexical formatives* (6.b) that appear in (5). Terminal symbols may have inherited and intrinsic attributes. Routines for lexical analysis and look-up replace the actual symbols in the input string by tokens of the corresponding lexical category and with the attribute frame associated with the category. At that stage, some degree of ambiguity is introduced, since it is often the case that a given input symbol may be associated with several parts of speech. The separation of lexical analysis from the parsing process is justified by the reduction it permits in the size of the control table used by the parser, and also by the fact that actual input symbols are analyzed by morphological routines much simpler than the general routines of the parser.

(6) a. **Lexical categories:**
N, A, V, P, Adv, Q, Art, Deg, M, BE1, BE2, HAVE, DO

b. **Specified formatives:**
for, that, whether, to

The grammar fragment of (5) yields derivation trees whose leaves are lexical categories, specified lexical formatives, or empty strings. The latter arise due to expansion of a non-terminal into the empty string. Each node in the derivation tree is annotated with the attribute frame of the symbol that labels the node. The attribution in the grammar may define or otherwise constrain the values of several of the attribute occurrences in the frame. Lexical analysis and look-up yields tokens with inherent attribute values; lexical insertion of the token under a leave of the tree is determined by the lexical insertion rule (7). Note that the intrinsic attribute values of the lexical category are determined by the lexical item inserted, while the inherited attributes of the category constrain the class of lexical items that may be inserted. This rule is the same as rule 2.1.(4) in Chapter 2. The attributes on the terminal nodes of the trees generated are then determined by the tokens produced by the lexical analyzer.

---

[1] Also, we introduced a slight modification of the rules that generate the English auxiliary system; cf. section 3.5.2.

(7) **Lexical Insertion Rule:**

$X \rightarrow t$

attribution:

$X.a \leftarrow t.a$, for each intrinsic attribute a of X

condition:

$X.a = t.a$, for each $a \in AI(X)$

The non-terminal vocabulary in (5) consists of the projections of the lexical categories (6.a), including the first and double bar projections XB and XP, the zero-level non-lexical categories C and I, and their projections IB, IP, CB, and CP. To the production set in (5) we add the production Z→CP, and we let Z be the start symbol of the grammar. Introduction of the start symbol Z is useful in sections 4.3 and 4.4 and has no other linguistic significance. Several productions in (5) rewrite a non-terminal into the empty string. The production NP→ϵ , for example, base-generates empty categories. As discussed in Chapters 2 and 3, we let the functional type of an empty category be determined by the context in which it appears.

In the grammar (5) no use is made of lexical features to analyze lexical categories. The more refined notation for grammar symbols is of no major interest here. We make limited use of Extended BNF conventions to collapse base rules and the savings that lexical feature cross-classification of grammar symbols would have, in terms of the number of "rules" that need to be written, are not significant. Also, the methods we employ to derive the analysis procedure apply to grammars whose symbols are atomic. Given that the space of categories defined by categorial features is finite, the same methods may be extended to work on complex symbols.

We assume the right-hand sides of the productions in (5) are ordered. As noted in Chapter 2, most current work in Government-binding assumes that the phrase-structure rules of a language are unordered, in the sense that they define an unordered context-free grammar (UCFG). The particular ordering effects that may be observed in a language are due to the interaction of language-specific parameters (e.g., head-first vs. head-final) and lexical properties of lexical items (e.g., directionality of Case and theta-role assignment). It is often assumed that the categorial status of specifiers, complements and adjuncts of a head are defined in a similar way by language-specific parameters and lexical properties of the head. The proposals which are currently available (Chomsky, 1981; Stowell, 1981; Travis, 1984; and several others) are, however, tentative and do not explain completely the ordering phenomena in a language, for

example the phenomena presented by (Jackendoff, 1977).[2] Thus we have opted for a fully explicit context-free production set to define the base. The grammar (5) is a fragment of the language specific production set for English that would result from the interaction of a minimal set of phrase-structure rules and other components of grammar (Stowell, 1981).

There is nothing in the present approach that precludes the use of a parametrized and modular approach to define the base. Whether native speakers learn or incrementally compile a expanded context-free form of the base for use in parsing, or whether the modular form is used in some direct way by the human parsing engine is a psycholinguistic issue which we expect would be very difficult to test. Either mechanism, pre-compilation or direct use, may be implemented computationally given an adequate modular definition of the base.

4.2.2 Some formal properties of the base

Before proceeding to the use of push-down automata as models for parsing, we discuss three formal properties of the grammar (5) which make it less than ideal for parsing, especially for "top-down" parsing. The three properties in question are ambiguity, left-recursion, and existence of identical prefixes in the grammar's production set. Even though those properties are properties of the grammar (5), they are also properties of the (expanded or modular) grammar of the natural languages which have been studied.

4.2.2.1 Ambiguity

A context-free grammar G is said to be *ambiguous* if for some string ω over its terminal alphabet it defines two different derivation trees. The grammar is said to be ambiguous of *degree k* if some such string has k distinct derivation trees. It is possible to write an ambiguous grammar for an unambiguous language. A (context-free) language is said to be *inherently ambiguous* if every (CF) grammar that describes it is ambiguous (Hopcroft and Ullman, 1978).

The grammar (5) is ambiguous, as shown by the example (8), where two distinct derivation trees are shown for the same terminal string. The PP shown may be attached as a complement of the noun *fish* (8.a), or as an instrumental adjunct of the verb *eat* (8.b). The attachement ambiguity (w.r.t. the base grammar (5)) may be resolved on semantic grounds, for example, as given by

---

[2] See also Chapter 2, section 2.3.5.

dictionary definitions of the meanings of the head words involved (*eat*, *fish*, and *bones*) and the relationships those definitions imply (Binot and Jensen, 1987). Another facet of the ambiguity of the string in (8), given our base grammar, is demonstrated in (9). The bracketing (9.a) shows the case where *We* is identified as the subject of the sentence, as expected; (9.b), on the other hand, shows an analysis where *We* appears in complementizer position, and an empty category in subject position (e.g., as due to movement from subject to complementizer position). Analysis (9.b) may be ruled out on extended syntactic grounds, if we take into account the attribute conditions imposed by the attribution in the grammar. In particular, the attribute conditions require that phrases appearing in C-specifier position be operators; these may be Wh-phrases or empty operators.

(8) a. We $[_{VB}$ate $[_{NP}$fish $[_{PP}$ with bones]]].

b. We $[_{VP}[_{VB}$ate [fish]] $[_{PP}$ with bones]].

(9) a. $[_{IP}$ We $[_{VP}$ate fish with bones]].

b. $[_{CP}$ We $[_{IP}$ $[_{NP} \epsilon]$ $[_{VP}$ate fish with bones]]].

Returning to (8), semantic factors are not always available to rule out an unlikely attachement like (8.b) of a PP. In the structurally parallel example (10), two derivation trees are available, and semantic factors do not decide between these.

(10) a. Bill [saw [the man $[_{PP}$ on the corner]]].

b. Bill [[saw [the man]] $[_{PP}$ on the corner]].

The use of ϵ-productions in (5) introduces another aspect of ambiguity that is even larger and more problematic. This is discussed in section 4.4.4.1 below, on attribute evaluation.

Given the ambiguity of the grammar (5), and the arguably inherent ambiguity of natural language in general, a complete parsing procedure for the grammar should be able to deliver all alternate analyses that the grammar defines for a given string. Ideally, the parsing procedure

should also rank the analyses produced by order of preference.[3] Heidorn (1976) describes a way of defining such a metric, based on Kimball's (1972) principles of surface structure parsing. Given that the English grammar is an attribute grammar, the observation of all-pass parsing should not be construed as advocating a model of parsing in which the context-free grammar operates in a first stage to produce derivation trees, while attribute annotation and evaluation apply in a second stage, after the derivation trees are available. As example (9) suggests, spurious analyses of the base may be filtered out, often at early stages, by carrying out structure building and attribute annotation and evaluation in parallel. We return to this point in section 4.4, when we discuss attribute evaluation.

4.2.2.2 Left recursion

A context-free grammar G is said to be *left-recursive* on category X if there is a derivation $Z \Rightarrow^* X\gamma \Rightarrow^* X\omega\gamma$, where γ and $\omega \neq \in$ are strings over the vocabulary of G. Left-recursive symbols lead to cycles in left-to-right, leftmost-first (LL) derivations, whose traversal can be interrupted only by examination of an arbitrarily large number of *lookahead* or right context) symbols (Waite and Goos, 1984). Left-recursion in context-free grammar G can always be eliminated by transformation of G to a *weakly* equivalent grammar G'. This is an option which we do not consider, since it would change completely the phrase-structure analyses that the grammar defines.

The grammar (5) is left-recursive due to presence of the production (11). This production permits NP specifiers in NP, as in (12).

(11) NP → NP NB

(12) a. [[The man]'s coat]

b. [[[The man]'s coat]'s pocket]

For LL parsers of the sort we develop here, left-recursion is especially damaging, since it eventually locks the parser into infinite derivation paths, as in (13). LL parsers require

---

[3] This is an overly ambitious goal, since the "preference" metric depends on complexity, syntactic, and semantic factors.

grammars in which left-recursion is absent, or some special mechanism to detect when they are pursuing an infinite derivation path. In section 4.5 we define the mechanism we use to avoid this problem in our LL algorithm.

(13) $Z \Rightarrow \ldots \Rightarrow NP\ \omega \Rightarrow NP\ NB\ \omega \Rightarrow NP\ NB\ NB\ \omega \Rightarrow \ \ldots \Rightarrow NP\ \ NB \ldots NB\ \omega$

4.2.2.3 Production prefixes

We say that two productions $X \rightarrow \alpha$ and $Y \rightarrow \beta$ share the same *prefix* $\gamma \in V^*$ if there exist strings δ, $\varepsilon \in V^*$ such that $\alpha = \gamma\delta$ and $\beta = \gamma\varepsilon$. The grammar (5) has several productions with identical prefixes. For example, the common prefix in the subset (14) of (5) is the symbol VB; some expansions of this symbol appear in (15).

(14)
VP → VB
VP → VB AdvP
VP → VB PP
VP → VB PP PP
VP → VB CP

(15)
VB → V
VB → V NP
VB → V CP

An LL parser must decide which production among those with identical left-hand sides should be applied at a given point in a derivation. The presence of the prefix VB in (14) makes it impossible to make the correct decision without some amount of lookahead. The strings generated by the symbol VB, as allowed by the expansions (15), can be arbitrarily long; hence, no fixed finite amount of lookahead is sufficient to discriminate among the productions in (14). Production prefixes are thus damaging to top-down parsers, although they are not as bad as left-recursion is, since they need not lead to endless derivation sequences. In section 4.5 we will consider one possible grammar transformation to eliminate the prefixes in (14).

4.3 Push-down Automata and Parsing Algorithms

One recurring theme in language theory is the identification of particular classes of languages with the automata that recognize them. Finite-state or Type-3 languages are be described by right-linear grammars or regular expressions and have the finite-state automata as their machine counterpart. The context-free or Type-2 languages are described by context-free grammars and have the push-down automaton as their machine counterpart. Understanding of the structure and capabilities of the push-down automaton (PDA) greatly simplifies the study parsing algorithms, by providing a primitive vocabulary and set of actions in terms of which all parsing actions may be described.

In this section we give a formal characterization of push-down automata and prove their equivalence to context-free grammars. The proof of equivalence is constructive; there are two construction procedures, which lead to distinct push-down automata and define two different styles of parsing. At the end of the section we look at the use of lookahead symbols to improve automaton performance and formulate a concrete algorithm with one symbol lookahead. We follow the notation and conventions of Waite and Goos (1984).

4.3.1 Push-down automata

A push-down automaton is an abstract device, which can be obtained from a finite-state automaton by adding a *push-down stack* as an additional storage structure. The PDA has an *input tape*, a *finite-state control*, and a *stack*. We let T be the input alphabet, Q the set of states of the finite-state control, and Γ the stack alphabet. The input tape contains the string of symbols to be recognized and the stack some string of symbols over the stack alphabet. If we write the stack contents on a horizontal line, the left-most symbol of the stack string is considered to be at the "top" of the stack. A PDA *configuration* is a triple $\sigma q \tau$, where $\sigma \in \Gamma^*$, $q \in Q$, and $\tau \in T^*$. If the PDA has reached configuration $\sigma q \tau$, we say the automaton is in state q, with stack contents σ, and τ is the unread portion of the input tape.

A PDA is a non-deterministic device. Depending on the first few top symbols of the stack, the state of the finite control, and the unread portion of the input tape, the finite state control makes available a number of possible moves. Of these, one is chosen to advance the automaton from the *current configuration* to the *next configuration*. Each move defines a string of symbols to

replace the top symbols of the stack, and a next state for the finite state control. Two types of moves are available. In the first, an input symbol is consumed; in the second, called an ϵ -move, no input symbol is consumed. The latter allow the PDA to manipulate the stack content without consuming an input symbol.

Given an initial PDA configuration $\sigma_0 q_0 \tau$, a sequence of moves may consume the input and lead to an accepting configuration $\sigma_f q_f \epsilon$. In this case we say that τ is a string in the *language* accepted by the automaton. There are two standard ways of defining an *accepting configuration.* According to the first, an accepting configuration is one in which the PDA stack is empty (i.e, $\sigma_f = \epsilon$); the language accepted is the set of all strings which, for some sequence of moves, consume the input and lead to an empty stack. The language defined is called *language accepted by empty stack.* An accepting configuration may also be defined by identifying a subset F of states as *final states.* The language accepted by the automaton is then the set of all strings which for some sequence of PDA moves consume the input and lead to a final state $q_f \in F$. The language defined is called *language accepted by final state.*

The two definitions of acceptance are equivalent, in the sense that they define the same family of languages, namely the context-free languages. The previous remarks are summarized in definition (16).

(16) A *push-down automaton* is a septuple $A = (T, Q, \delta, q_0, F, \Gamma, \sigma_0)$, where

(i) T is the input alphabet,

(ii) Q is a finite set of states,

(iii) Γ is the stack alphabet,

(iv) $q_0 \in Q$ is the initial state,

(v) $F \subseteq Q$ is the set of final states,

(vi) $\sigma_0 \in \Gamma \cup \{\epsilon\}$ is the initial stack symbol (if $\sigma_0 = \epsilon$, the initial stack is empty),

(vii) δ is a many-many mapping from $\Gamma^* \times Q \times T^*$ into itself. Each element $\sigma q\tau \rightarrow \sigma' q'\tau'$ of δ defines a possible move of the PDA. If $\tau = a\tau'$, the move consumes the next input symbol $a \in T$; if $\tau = \tau'$, the move is an ϵ-move.

A transition element $\sigma q\tau \rightarrow \sigma' q'\tau'$ in δ is interpreted as follows: if $\sigma\eta q\tau\mu$ is the current PDA configuration, where $\sigma\eta$ is the current stack and $\tau\mu$ the unread input, the element may be applied to advance the automaton to the configuration $\sigma'\eta q'\tau'\mu$. The *move* relations $\Rightarrow$ and $\Rightarrow^*$ of a PDA are defined in (17).

(17) Let $A = (T, Q, \delta, q_0, F, \Gamma, \sigma_0)$. Given PDA configurations $\sigma\eta q\tau\mu$, $\sigma'\eta q'\tau'\mu \in \Gamma^* \times Q \times T^*$, we write $\sigma\eta q\tau\mu \Rightarrow \sigma'\eta q'\tau'\mu$ if there exists a transition element $\sigma q\tau \rightarrow \sigma' q'\tau'$ in δ. The relation $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$.

We will say that the PDA A accepts an input string τ if starting from the initial configuration $\sigma_0 q_0 \tau$ it consumes the input string, empties the stack, and enters some final state f. The notions of acceptance and language accepted by a PDA are defined in (18).

(18) The PDA A *accepts* an input string $\tau \in T^*$ if $\sigma_0 q_0 \tau \Rightarrow^* \epsilon f \epsilon$, for some state $f \in F$. The *language* accepted by the PDA is the set $L(A) = \{\omega : \sigma_0 q_0 \omega \Rightarrow^* \epsilon f \epsilon$, for some $f \in F\}$.

Definition (16) of push-down automaton follows that of Waite and Goos (1984), and is different from others found in the literature (Hopcroft and Ullman, 1978). The elements σ and σ' of the transitions in δ are members of Γ^* rather that Γ; hence, the automaton may rewrite several symbols of the stack in one move, and could be called a *generalized push-down automaton.* The definition given here is more appropriate for our present purposes; it permits direct statement of the two parsing models we shall consider. Note also that definition (18) of acceptance is by empty stack and final state.

4.3.2 Relation of push-down automata to context-free grammars

We prove the equivalence theorems between context-free languages and push-down automata.

Theorem 1: For every context-free grammar G, there exists a PDA A such that L(A) = L(G).

Proof: The proof of the theorem is constructive. Given the context-free grammar G, a method is given for constructing the push-down automaton A. There are two construction procedures, which lead to distinct automata and parsing models. These are the so-called *top-down* and *bottom-up* analysis methods. We consider each construction in turn.

Construction 1 (Top-down analysis): Let G = (N, T, P, Z). We define the automaton $A = (T, \{q\}, \delta, q, \{q\}, T \cup N, Z)$, where $\delta = \delta_1 \cup \delta_2$ is the mapping (19).

(19) $$\delta = \{ Xq\epsilon \rightarrow x_1 \ldots x_n q\epsilon : \quad X \rightarrow x_1 \ldots x_n \in P, \ n \geq 0 \} \cup \{ tqt \rightarrow \epsilon q\epsilon : \ t \in T \}$$

In this construction, the initial stack is Z, the start symbol of G. The automaton performs *top-down* or predictive analysis, tracing left-most derivations from the axiom Z, while the input is consumed from left-to-right. At each step in a derivation, if $X \in N$ is the top symbol of the stack and $X \rightarrow x_1 \ldots x_n$ is a production in P, the automaton may use the corresponding transition $Xq\epsilon \rightarrow x_1 \ldots x_n q\epsilon$ in δ_1 to replace X by the right-hand side symbols in the production, thus predicting a sequence of symbols and phrases that should be present in the unconsumed portion of the input string. This transition is an ϵ-move, since it does not consume an input symbol. If the top symbol of the stack is some terminal t and the next input symbol is $t' = t$, the automaton may use the corresponding transition in δ_2 to consume t' from the input stream and pop t off the stack.

A *stack invariant* of this automaton is that at each step in a derivation the stack contains a string $\sigma \in (N \cup T)^*$ such that σ may be used to derive the unconsumed section of the input string: If $\sigma q \tau$ is the current configuration, then $\sigma \Rightarrow^* \tau$.

Construction 2 (Bottom-up analysis): Let G = (N, T, P, Z). Now we may define the automaton $A' = (T, \{q\}, \Delta, q, \{q\}, T \cup N, \epsilon)$, where $\Delta = \Delta_1 \cup \Delta_2 \cup \Delta_3$ is the mapping (20).

(20) $$\Delta = \{ x_1 \ldots x_n q\epsilon \rightarrow Xq\epsilon \ : X \rightarrow x_1 \ldots x_n \in P, \ n \geq 0 \}$$
$$\cup \{ \epsilon qt \rightarrow tq\epsilon : \ t \in T \} \cup \{ Zq\epsilon \rightarrow \epsilon q\epsilon \}$$

Starting with an empty stack, this automaton traces right-most derivations of the input string by consuming terminal symbols from it and grouping them into successively larger phrases, until it

arrives at the axiom Z of the grammar. An input symbol t may be consumed by applying the corresponding transition $\in qt \rightarrow tq\in$ in Δ_2; this transition copies t on top of the previous PDA stack. If $x_1 \ldots x_n$ are the top n elements of the stack and $X \rightarrow x_1 \ldots x_n$ is a production of P, the transition $x_1 \ldots x_n q\in \rightarrow Xq\in$ in Δ_1 may apply to reduce the n symbols to the single symbol X. When the stack contains the single symbol Z and the input has been consumed, the transition in Δ_3 may apply to empty the stack. At this point we have acceptance of the input string.

The *stack invariant* associated with this automaton is that at any step the content of the stack is a string $\sigma \in (N \cup T)^*$ which derives the portion of the input string that has already been consumed: If $\omega = \tau\eta$ is the input string and $\sigma q \eta$ is the current configuration, then $\sigma \Rightarrow^* \tau$.

The proof of the theorem is completed by showing that the language accepted by the constructed automaton in either case is equal to the language generated by the grammar -- i.e., $L(A) = L(A') = L(G)$. We outline the proof for the first construction, and omit the proof for the second.

Let $\tau \in L(G)$. Then there is a left-most derivation $Z = \sigma_1 \Rightarrow \sigma_2 \Rightarrow \ldots \Rightarrow \sigma_k = \tau$ of τ. But given this derivation we can immediately construct a sequence of PDA moves which accept τ. Starting with the configuration $Zq\in$, if the top symbol of the stack is $X \in N$, apply the transition corresponding to the production used at the i-th step in the derivation; if the top symbol is a terminal t, it matches the next input symbol and may be consumed. Hence $\tau \in L(A)$ and $L(G) \subseteq L(A)$. Conversely, if $\tau \in L(A)$, there is a sequence of PDA moves which accept τ, from which the left-most derivation L may be obtained. Hence $\tau \in L(G)$, $L(A) \subseteq L(G)$, and we may conclude $L(G) = L(A)$. □

The constructions of theorem 1 may be applied to an arbitrary context-free grammar G. Hence we conclude that the family $\mathbf{L}_{CFG}$ of languages generated by context-free grammars is contained in the family $\mathbf{L}_{PDA}$ of languages accepted by PDAs. That is, $\mathbf{L}_{CFG} \subseteq \mathbf{L}_{PDA}$. We state without proof the converse of this statement.

Theorem 2: $\mathbf{L}_{PDA} \subseteq \mathbf{L}_{CFG}$.

The reader is referred to Hopcroft and Ullman (1978) for proof of this theorem.

4.3.3 Deterministic parsing and lookahead

The two constructions of Theorem 1 may be used to construct a PDA that recognizes the language generated by an arbitrary context-free grammar. In the general case, however, the PDAs obtained are non-deterministic and need not even lead to a recognition algorithm, since the sequence of moves that they may follow for some inputs need not terminate. An automaton is deterministic if the current configuration uniquely determines the next transition which may be made (21). In this section we state the conditions under which the constructions of theorem 1 yield a deterministic automaton and also consider an extension of the top-down construction procedure, which uses lookahead symbols in the unparsed portion of the input string to help select the next PDA transition.

(21) Let $A=(T, Q, \delta, q_0, F, \Gamma, \sigma_0)$ be a PDA. A is said to be *deterministic* if for every configuration $\sigma q\tau$ there is at most one transition element $\sigma' q\tau' \rightarrow \sigma'' q'\tau''$ in δ which applies (i.e., such that $\sigma=\sigma'\eta$ and $\tau=\tau'\mu$, for some $\eta \in \Gamma^*$ and $\mu \in T^*$).

It is easy to check that for the context-free grammar (5) the top-down construction of the theorem does not yield a deterministic automaton. For example, if the current automaton configuration is $NP\sigma q\tau$ (the PDA is beginning parsing of a noun phrase), there are ten different transitions which may be applied, corresponding to the ten productions for NP in the grammar.

Unfortunately, it is not possible to sharpen either of the constructions in theorem 1 (or any other conceivable construction) so that the resulting automaton is always deterministic. Unlike regular languages, which can always be analyzed deterministically (Hopcroft and Ullman, 1978), there are context-free languages for which no deterministic procedure exists. Only by imposing additional restrictions on the grammar can it be guaranteed that a deterministic automaton will result from a given PDA construction procedure. The restrictions required upon the grammar depend upon the construction procedure. The class of grammars for which the top-down construction yields a deterministic PDA may be enlarged by use of lookahead symbols, as detailed below. For grammars which fall outside the new class, including (5), the improved construction method can be used to produce automatons which, although non-deterministic, perform much better that the simple automatons of construction 1.

Let $V=N \cup T$ be the vocabulary of a grammar G and let # be a symbol not in V. This symbol will be used to mark the right end of strings over V. Let $\omega \in V^*$. We use $|\omega|$ to denote the

length of ω. For $k \geq 0$, the *k-head* of ω, written $k{:}\omega$, is the substring of ω consisting of its first k symbols, if $|\omega| \geq k$, or $\omega\#$, if $|\omega| < k$. $\text{FIRST}_k(\omega)$ is the set of all k-heads of terminal strings derivable from ω. $\text{FOLLOW}_k(\omega)$ is the set of all k-heads of terminal strings which may follow ω. Formally, given a string ω we have the definitions (22).

(22) a. $k{:}\omega =$ if $|\omega| \geq k$ then α else $\omega\#$, where $|\alpha| = k$ and $\omega = \alpha\eta$

b. $\text{FIRST}_k(\omega) = \{ \alpha : \alpha = k{:}\eta$, for some $\eta \in T^*$ such that $\omega \Rightarrow^* \eta \}$

c. $\text{FOLLOW}_k(\omega) = \{ \alpha : \alpha \in \text{FIRST}_k(\eta)$, for some $\eta \in V^*$ such that $Z \Rightarrow^* \nu\omega\eta \}$

The class of grammars for which the first construction of Theorem 1 gives a deterministic PDA is the class of LL(0) grammars, as in (23).

(23) A context-free grammar $G = (N, T, P, Z)$ is *LL(k)*, for $k \geq 0$, if given arbitrary derivations

$Z \Rightarrow_L \mu X\xi \Rightarrow \mu\nu\xi \Rightarrow^* \mu\gamma$, and

$Z \Rightarrow_L \mu X\xi \Rightarrow \mu\omega\xi \Rightarrow^* \mu\gamma'$,

where $\mu, \gamma, \gamma' \in T^*$, $X \in N$, and $\nu, \omega, \xi \in (N \cup T)^*$, the following condition holds: $k{:}\gamma = k{:}\gamma'$ implies $\nu = \omega$.

The terminal strings γ and γ' in (23) may be derived from $\nu\xi$ and $\omega\xi$, respectively. Since $X \rightarrow \nu$ and $X \rightarrow \omega$ are productions of G, it follows that $X\nu \Rightarrow^* \gamma$ and $X\nu \Rightarrow^* \gamma'$. The LL($k$) condition simply requires the k-heads of γ and γ' be sufficient to predict which of the candidate productions for X should be applied in the next step of the derivation. The up to k symbols in $k{:}\gamma$ and $k{:}\gamma'$ are called the *lookahead symbols* used by the predictor. If $k = 1$, we omit the subscript k from the sets FIRST_k and FOLLOW_k. It can be proved that for every LL(k) grammar G there exists a deterministic PDA A such that $L(A) = L(G)$. The automaton recognizes each sentence of L(G) by reading it from left-to-right, producing a left-most derivation, and examining at most k input symbols at each derivation step.

To illustrate the use of lookahead symbols in top-down analysis, let $k = 1$ and consider the four productions NP→NB, NP→ArtP NB, NP→QP NB, and NP→AP NB in the grammar (5). Suppose now that the automaton is currently trying to parse an NP and that the next symbol in the input string is *the* -- in short, the current configuration is $\text{NP}\sigma q$ *the* τ, for some $\sigma \in \Gamma^*$ and τ

$\in$ T*. Given the lookahead symbol *the*, the predictor can discard three of the productions among the four candidates, and decide that the production to apply at this point is NP→ArtP NB. If the remaining six productions for NP in (5) are included, however, the single lookahead symbol is not enough to distinguish among the ten alternatives; it is enough only to reduce the initial choice to five. This single fact is enough to see that the base grammar (5) is not LL(k), for $k=1$. In an even more fundamental way, the presence of the left-recursive production NP → NP NB implies that the grammar is not LL(k), for any k. This follows from theorem 3, which is stated without proof. It follows that the grammar (5) does not have a deterministic top-down parser. Providing some amount of lookahead, though, has the positive effect we just noted.

Theorem 3: An LL(k) grammar cannot have any left-recursive non-terminals.

We now develop a construction procedure which, given an arbitrary CFG G, yields a PDA which does top-down analysis using k symbols of lookahead. If the grammar is LL(k), the PDA produced is deterministic. The construction is taken from Waite and Goos (1984).

Let G=(N, T, P, Z) be the context free grammar and assume that the productions in P are numbered 1, ..., m. Unlike the automaton of construction 1 in theorem 1, the new automaton has more than one state. Each state is a triple (p, j, ω), where p is the number of production $X \rightarrow x_1 \ldots x_{n_p}$, $1 \leq j \leq n_p$, and ω is the set of k-heads of terminal strings which may follow strings derived from X, *in the context in which* X *is introduced.* Thus $\omega \subseteq \text{FOLLOW}_k(X)$. Each such triple is called a situation and can be written in the more descriptive form (24). The dot (•) used in a situation is a symbol outside the vocabulary of G and indicates how far the analysis has proceeded along the production's right-hand side.

(24) $\quad [X \rightarrow \mu \bullet \nu : \omega\ ]$, where $\mu = x_1 \ldots x_j$ and $\nu = x_{j+1} \ldots x_{n_p}$

Now let the stack alphabet of the automaton be $\Gamma = Q$. Then δ is a function on $Q^* \times Q \times T^*$. The set of states Q and transitions in δ are defined inductively by the procedure (25).

(25) Construction procedure for LL(k) parser

1. Let $q_0 = [Z \rightarrow \bullet CP : \{\#\}]$, $Q = \{q_0\}$, $\sigma_0 = q_0$, and $\delta = \{\}$. Note that q_0 is the initial state and also the initial stack content; # is the end-of-string marker. Also, let $F = \{q_0\}$, so that the automaton accepts if q_0 is reached with an empty stack.

2. Suppose $q = [X \rightarrow \mu \bullet v : \omega]$ is an element of Q not yet considered.

   (i) If $v = \epsilon$, then let $\delta := \delta \cup \{ q'q\epsilon \rightarrow \epsilon q'\epsilon : q' \in Q \}$.

   (ii) If $v = t\gamma$, for some $t \in T$ and $\gamma \in V^*$, then let $q' = [X \rightarrow \mu t \bullet v : \omega]$ and set
   $Q := Q \cup \{ q' \}$,
   $\delta := \delta \cup \{ \epsilon qt \rightarrow \epsilon q'\epsilon \}$

   (iii) If $v = B\gamma$, for some $B \in N$ and $\gamma \in V^*$, then let $q' = [B \rightarrow \mu B \bullet v : \omega]$,
   $H = \{ [B \rightarrow \bullet\beta : FIRST_k(\gamma\omega)] : B \rightarrow \beta \in P \}$, and set
   $Q := Q \cup \{ q' \} \cup H$,
   $\delta := \delta \cup \{ \epsilon q\tau \rightarrow q'h\tau : h \in H, \text{ and } \tau \in FIRST_k(\beta\gamma\omega) \}$.

3. Repeat step (2) until all elements of Q have been considered.

The construction procedure terminates for every grammar, since the set of situations that must be considered is finite. The test $\tau \in FIRST_k(\beta\gamma\omega)$ in step 2.iii is the test on the k lookahead symbols τ. It can be shown that for a given grammar G the resulting automaton is deterministic if and only if G is LL(k).

4.3.4 Top-down algorithm with one-symbol lookahead

A practical parser based on the top-down PDA construction procedure of the previous section is usually limited to one symbol lookahead. The main reason for this limit is the sharp increase in the sizes of the set of states Q and finite-state transition function δ of the PDA generated once more lookahead symbols are used.

Assuming a one symbol lookahead limit and that the grammar is LL(1), it is possible to change the third alternative in step 2 of (25) to the simpler (26), while guaranteeing that the resulting PDA will be deterministic. Note that the subscripts from the sets $FIRST_k$ and $FOLLOW_k$ have been droped, since $k=1$.

(26) If $v=B\gamma$, for some $B \in N$ and $\gamma \in V^*$, then let $q'=[B\rightarrow\mu B\bullet v : \omega\ ]$,
$H=\{\ [B\rightarrow\bullet\beta : FOLLOW(B)\ ]:\quad B\rightarrow\beta \in P\ \}$, and set
$Q := Q \cup \{q'\} \cup H$,
$\delta := \delta \cup \{\ \epsilon q\tau\rightarrow q'h\tau :\ h \in H,$ and $\tau \in FIRST(\beta FOLLOW(B))\ \}$.

This change in the procedure has the effect of further reducing the number of states and size of δ in the PDA obtained. Given a production $p=B\rightarrow\beta$, the set $W(p)=FIRST(\beta FOLLOW(B))$ is called the *director set* of p. This set is independent of any derivation and may be computed from the symbols in the grammar alone. The reader is refered to Waite and Goos (1984) for a recursive procedure to compute the sets FIRST(X) and FOLLOW(X) for each symbol X in the vocabulary of the grammar, and the director set $W(p)$, for each production. A version of this procedure has been efficiently implemented as a Prolog program. The LL(1) property on a grammar may be tested in a direct way, by reference to the director sets associated with its productions, as in Theorem 4.

Theorem 4: A context-free grammar is LL(1) if and only if for every pair of distinct productions $p=X\rightarrow\gamma$ and $p'=X\rightarrow\gamma'$, the director sets $W(p)$ and $W(p')$ are disjoint.

By theorem 4, given PDA configuration $\sigma q\tau$, where $q=[\ X\rightarrow\mu\bullet X\gamma : \omega\ ]$, the director sets for the productions of X may be used to predict the production that should be used to expand X; the production $p=X\rightarrow\beta$ may be applied only if $FIRST(\tau) \in W(p)$.

A fairly direct Prolog implementation[4] of the LL(1) PDA of construction (25)-(26) is given in (28) below. The program assumes two three-place logical predicates *ll1_parse* and *ll1_production*, as shown in (27). The arguments σ, q, and ω in (27.a) define a possible PDA configuration $\sigma q\omega$. The predicate holds true of a given triple of arguments just in case an accepting configuration may be reached from the current configuration defined by the three arguments. The arguments

---

[4] The Prolog language used is VM/Prolog.

X and β in (27.b) correspond to a production $p = X \rightarrow \beta$, and W to the precomputed director set W(p). As given by construction (25), states and stack symbols are situations. A situation [$X \rightarrow \mu \bullet \nu$: ω] of the PDA is represented as the term *sit*(X, μ, ν) in the parser. Notice that the right context ω has been omited, since in the construction (25) it is needed only in step 2 by the predictor. Here we assume (26) in place of (25.2.iii) for the predictor, and the director sets W(p) are precomputed.

(27) a. lll_parse(σ, q, ω)

b. lll_production(X, β, W)

(28) **Prolog Implementation of LL(1) Pushdown Automaton**

```
/* 1. Complete parsing of production */
lll_parse( [ Top| StackTail], sit( _, _, []), RestOfInput) ←
    lll_parse( StackTail, Top, RestOfInput).

/* 2. Consume terminal 'Tok' in input string */
lll_parse( Stack, sit( L, R, [Tok| RTail]), [ Tok| RestOfInput]) ←
    append( R, [ Tok], Rnew]) &
    lll_parse( Stack, sit( L, Rnew, RTail), RestOfInput).

/* 3. Expand recursively non-terminal 'B' */
lll_parse( Stack, sit( L, R, [B| RTail]), [ Lookahead| RestOfInput]) ←
    lll_production( B, RHS, W) &
    member( Lookahead, W) &
    append( R, [ B], Rnew]) &
    lll_parse( [ sit( L, Rnew, RTail)| Stack],
                sit( B, [], RHS), [ Lookahead| RestOfInput] ).

/* 4. Termination condition */
lll_parse( [], sit( "Z", _, _ ), [ "#"]).
```

Assuming Z → α is the start production, operation of the PDA for parsing an input string τ is started by giving the goal (29).

(29) ← lll_parse([sit("Z", [], α)], sit("Z", [], α), τ #).

If the grammar is LL(1), the parser is deterministic and terminates with success if and only if τ is in the language generated by G. The parser is deterministic in the sense that once a production is selected for expansion of a non-terminal, this decision is not retracted -- i.e., there is no backtracking behind the *lll_parse* predicate. But notice that backtracking is used in clause 3 of the parser for selection of the next production that should be tried. This backtracking is minor and could be eliminated in a more efficient implementation. The modification required is indexing from the lookahead symbol into the production set, based on the director sets of the productions. If G is LL(1), the next input symbol and the category on top of the stack select at most one production for the next step in the derivation.

The LL(1) automaton's stack is represented explicitly as the first argument of the *lll_parse* predicate. This stack could also be encoded implicitly in the stack frames corresponding to recursive procedure calls in the program. Having the stack explicitly as an argument permits easy statement of the stack constraints that we will add to the basic LL(1) algorithm (28), in section 4.5.

4.4 The Attribute Component of the Grammar

In this section we look at the attribute component of the grammar developed in Chapter 3. In section 4.4.1, we summarize the association between attributes and syntactic categories and define a concrete representation for attribute values, attributes, and attribute frames. In section 4.4.2 we classify the attribution rules and conditions in the grammar according to the Government-binding subtheory to which they belong. This helps to visualize the correspondence between the attribute grammar and Government-binding theory. In section 4.4.3 we study the attribute dependencies induced by the attribution. In particular, we study the dependencies involved in node enumeration, government, Case-marking, and θ-marking. Lastly, in section 4.4.4 we define an attribute evaluation scheme and formulate a procedure for constructing an attribute evaluator from the grammar. The attribute evaluator procedures are applied in the same phase as LL(1) parser actions, to implement an "attribute directed" parser.

4.4.1 Attribute system and representation

The attribute grammar of Chapter 3 uses nearly 40 attributes names, many of which define attributes associated with several categories in the grammar. The meaning of each attribute, its use, and domain are defined in Chapter 3, along with an indication of whether the attribute is inherited or synthesized when associated with a given category in the grammar.

The attribute grammar is formally a quadruple (G, A, R, B). For each symbol X in the vocabulary of the underlying grammar G, a set or *attribute frame* A(X) of attributes is defined. Each attribute in A(X) is written X.*a*, where *a* is the name of the attribute. When it is clear or irrelevant what the category X is, we omit it from the print-name of the attribute and write *a* in place of X.*a*. If *a* is an attribute associated with X and Y it is possible that X.*a* is inherited while Y.*a* synthesized, or viceversa. The association of attributes with syntactic categories in G is shown in (30.a-c) below. We show only those attributes which are more linguistically relevant; we omit, for example, the attributes *node* and $node_0$, which are associated with all categories in the grammar.

(30.a)

Lexical Category	**Attribute Frame**
N	[*AGR, Case, Wh, pronominal, anaphoric, proper*]
V	[*subcat, tense, AGR, Case,* Θ_I, θ_E, *ecm, tense, Wh, mood, part, perf, pass*]
P	[*subcat, Case*]
A	[*AGR*]
Adv	[ ]
Art	[*AGR, Wh*]
Q	[*AGR, Wh*]
Deg	[ ]
M, HAVE, BE1, BE2, DO	[*AGR, tense, mood, part, perf, pass*]

(30.b)

Category	Attribute Frame
NP	[*empty, Gov, head, AGR, Case, Wh, θ pronominal, anaphoric, proper, RefIndex, Chain, A-Chain, AB-Chain, PAS,* PAS_S, *AAS*]
NB	[*AGR, Case,* wh, head, *pronominal, anaphoric, proper, A-Chain, AB-Chain, PAS,* PAS_S, *AAS*]
VP, VB	[*tense, AGR,* θ_E, *head, A-Chain, AB-Chain, PAS,* PAS_S, *AAS, mood, part, perf, pass*]
PP	[*head, θ, A-Chain, AB-Chain, PAS,* PAS_S, *AAS*]
PB	[*head, A-Chain, AB-Chain, PAS,* PAS_S, *AAS*]
AP	[*AGR, θ*]
AB	[*AGR*]
AdvP	[head, *θ*]
AdvB	[head]
ArtP, ArtB	[*AGR, Wh*]
QP, QB	[*AGR, Wh*]
DegP, DegB	[ ]

(30.c)

Category	Attribute Frame
CP	[*tense, Wh, θ, head, Gov, Case, ecm, A-Chain, AB-Chain, PAS,* PAS_S, *AAS*]
CB	[*tense, Wh, head, Gov, Case, ecm, A-Chain, AB-Chain, PAS,* PAS_S, *AAS*]
C	[*tense,* $tense_m$, *Wh, AGR, Case, comp, mood, part, perf, pass*]
IP	[*tense, AGR, head, Gov, Case, ecm, A-Chain, AB-Chain, PAS,* PAS_S, *AAS, mood, part, perf, pass*]
IB	[*tense, AGR,* θ_E, *head, A-Chain, AB-Chain, PAS,* PAS_S, *AAS, mood, part, perf, pass*]
I	[*tense,* $tense_m$, AGR_S, AGR_m, *mood, part, perf, pass,* $mood_m$, $part_m$, $perf_m$, $pass_m$]

Some attributes in (30.a), associated with lexical categories, are inherited by the categories, while others are intrinsic. The inherited include, for example, N.*Case* and V.*subcat*, which depend on the context in which the category appears. The intrinsic attributes include V.*Case* and V.θ_E, which are determined by the verb inserted under the category. A lexical item η of category X

has an attribute set $A(\eta)=A(X)$ of intrinsic attributes associated with it. Insertion of η under the category X is possible only if the values of the attributes in $A(\eta)$ match those which are already instantiated in the set A(X), as given by the lexical insertion condition in (7) above. Unification is the operation used to implement the attribution rule and condition in (7), as detailed in section 4.4 on attribute evaluation.

The domains of most attributes in (30) are finite sets of small cardinality, consisting of atomic values. This is the case of the attributes *Case* and θ, and of the binary-valued attributes *tense*, *ecm*, *Wh*, *pronominal*, *anaphoric*, *mood*, etc, as shown in (31). The Prolog representation of a value α in these domains is a constant whose print-name is α. The domain of several other attributes in the grammar, including *node*, *head*, *Chain*, *A-Chain*, *AB-Chain*, *RefIndex*, etc, is the natural numbers, including zero. These are represented as Prolog small integers.

(31) a. $dom(Case) = \{ Nom, Acc, Dat, Gen, nil \}$

b. $dom(\theta) = \{ agent, theme, goal, ..., nil \}$

c. $dom(x) = \{+, -\}$, for x= *tense*, *ecm*, *Wh*, etc.

In general, the domain of an attribute is a collection of ground functional terms $f(t_1, ..., t_n)$, where f is an n-ary functor, $n \geq 0$, and the t_i are terms. When $n=0$ we say the term is atomic or a constant. For $n > 0$, the most important functor is the infix list constructor "." , which is used in the representation of sets, stacks, and lists. A dotted pair $\alpha.\beta$ is a list whose head is α and whose tail is β, and may also be written $[\alpha| \beta]$; *nil* denotes the empty list or []. Attributes whose domains are non-atomic valued include *subcat*, *AAS*, *PAS*, and Θ_I . *subcat* is a list of categories, *AAS* and *PAS* sets of integers, corresponding to *node*-numbers, and Θ_I a list of "internal" θ-roles from (31.b).

We represent each attribute a in the grammar as a term $a(v)$, where v is a logical variable that may be substituted for the attribute's value. This representation is a direct encoding of an attribute-value pair $[a \ v]$, in which the attribute name is represented by the function symbol and the attribute value by the logical variable. In the usage below, when we refer to "the value of an attribute a", we should understand "the value of the variable v associated with a." If X is a symbol of the grammar, the attribute set A(X) associated with X is represented by the frame (32) of functional terms, where $a_1, ..., a_k$ are the attributes in A(X). The order of the attribute

occurrences in the frame (32) is fixed, so it is possible to access the attributes by the position they occupy. The frame representation (32) could be optimized by omiting attribute names from it, and using the attribute's position alone to identify the attribute. However, the readability of the logic program we develop is enhanced by retaining the attribute names in the representation.

(32) $[a_1(v_1), ..., a_k(v_k)]$

For simplicity in the implementation of the parser, we let a copy of the attribute frame (32) be attached to each node labelled X in a derivation tree. In this manner all attribute occurrences of the tree are represented. This is not a necessity; it may be possible to use global storage cells or stacks to store the value of some of the attributes used in the computation, but not needed after the parse is built. Optimization of the use of attribute storage is an implementation question not addressed here; see (Sonnenschein, 1984) and (Kastens, Hutt, and Zimmermann, 1982) for approaches to the problem. The benefits of this optimization may be ignored in this parser, since given the range of natural language inputs expected (strings consisting of fewer than, say, 100 words) the fully annotated tree structures computed by the parser are of modest size, and hence able to fit in the computer's physical address space. This kind of optimization becomes imperative for compilers for programming languages, where the size of typical inputs (programs) is in the order of tens of thousands of tokens.

4.4.2 Correspondence of the attribution and Government-binding

The principal attribution rules and conditions of the attribute grammar were developed in Chapters 2 and 3. In the format there, the grammar appears rather monolithic, since attribution rules and conditions associated with a given production are listed immediately following the production. It is common to use this format for the statement of attribute grammars (Waite and Goos, 1984); it is deemed advantageous since it makes clear all the "semantic actions" and conditions that are associated with each production. The format saves space in the statement of the grammar and, more importantly, makes evident all actions and tests that the attribute evaluator must perform during operation of the parser. The format has the disadvantage of making less than clear the correspondence between the grammar which is stated and the possibly modular definition of the language on which the grammar is based. In our case, the format shown does not make perspicuous that relation between the grammar and the original Government-binding theory on which we based our work. The relation between the two can be elucidated as follows.

Each attribute in the grammar is associated with and contributes to the implementation of one particular subtheory of the Government-binding theory, as shown in (33). This association is clear in Chapter 3, where realizations of the GB subtheories were systematically developed. To implement the theory of movement, it was assumed that certain other components of the GB theory had been implemented (e.g., government, *Case*, and theta-marking) and certain other components were ignored (e.g., control, and the rules for the auxiliary system), since they were not relevant at that stage. We proceeded in this fashion to derive the rules that implement the subtheories of government, Case, thematic relations, move-α, the auxiliary system, and finally Binding. The subtheory of control and the derivation of Logical Forms were omitted. Bounding theory was subsummed under movement, since the Chain rule takes into account the constraints that Subjacency imposes on the original movement rule.

(33)

GB subtheory	**Attributes**
X-bar theory	None; we do not use categorial features
θ-theory	θ, θ_E , Θ_i
Case theory	*Case*, *ecm*
Binding theory	*RefIndex*, *AAS*, *PAS*, PAS_S
Bounding theory	None; subsumed under move-α
Control theory	*RefIndex*
Government	*Gov*, *head*, *node*
move-α	*Chain*, *A-Chain*, *AB-Chain*, *Wh*, *empty*
Auxiliary	*tense*, *mood*, *part*, *perf*, *pass*
Lexicon	*AGR*, *subcat*, *pronominal*, *anaphoric*, *proper*, *count*

The association (33) between attributes and GB subtheories may be extended to attribution rules and conditions. Intuitively, if a is an attribute associated with subtheory T, then the attribution rules that define the value of a are also associated with T. Each attribution rule $X.a \leftarrow f(\ldots)$ in $R(p)$, for some production p, is part of the implementation of the subtheory T. The attribute occurrences on the right-hand side of the attribution rule represent interactions between theory T and other subtheories in Government-binding. Similarly, if an attribute condition constrains the value of $X.a$, then that condition is also associated with theory T.

4.4.3 Attribute dependencies

In order to compute the value of an attribute it is necessary to know the values of all attributes

upon which the attribute depends.[5] For a given annotated structure tree, these dependency relations can be shown as a directed graph, in which the nodes represent attributes, and the directed edges represent the dependency relations between attributes. For example, the first attribution rule in (34) induces a dependency of the attribute X.*a* on the attribute X.*b*. Corresponding to the attribution rules in (34) we obtain the dependency graph (35).

(34) $X \rightarrow Y\ Z$

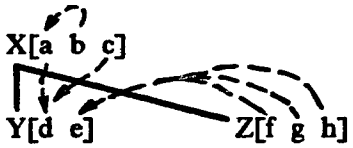
attribution:

$X.a \leftarrow X.b$

$Y.d \leftarrow [X.a, X.c]$

$Y.e \leftarrow$ if $Z.f$ then $Z.g$ else $Z.h$

(35)



We consider now the principal attribute dependencies of the grammar in Chapter 3. We begin with the attribution rules that define the values of the attributes *node* and $node_0$ in trees. These two attributes do not depend on any other attribute in the grammar and have little linguistic significance; their sole purpose is to enumerate the nodes in each derivation tree. Each production in the grammar, including lexical insertion, satisfies one of the schema in (36); the attribution for *node* and $node_0$ is shown immediately following the schema. Thus *node* is an inherited attribute, $node_0$ is synthesized, and both attributes may be evaluated "from left-to-right" in the sense of (Bochmann, 1976).

(36) a. $X_0 \rightarrow X_1\ X_2 \cdots X_n$

attribution:

$X_1.node \leftarrow X_0.node + 1$

$X_{i+1}.node \leftarrow X_i.node_0$, for $i = 1, \ldots, n\text{-}1$

$X_0.node_0 \leftarrow X_n.node_0$

---

5 This condition assumes *value* semantics. It is possible to relax this condition if we assume *lazy* semantics, where the value of an attribute need not be evaluated until it is actually needed.

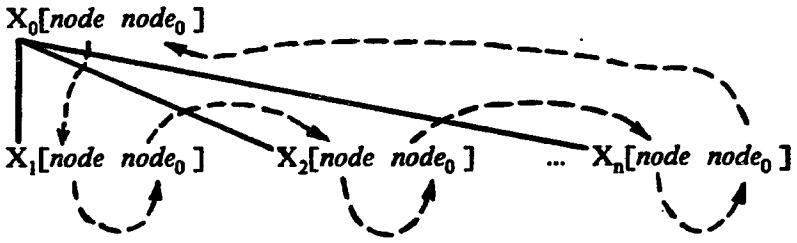
b. $X \rightarrow \in \mid t$, for t a specified or lexical formative

attribution:

$X.node_0 \leftarrow X.node + 1$

The dependency graph that arises from the attribution in (36) is shown in (37). The dashed edges at the bottom of the graph show the dependency of $node_0$ on *node* induced by (36.b), for a given X_i, and they are present just in case X_i is expanded into the empty string or a formative t is inserted under it. The dependency graph (37) is used in section 4.4.4 in the derivation of the attribute evaluation scheme.

(37)



We now consider the dependencies induced by three core grammatical processes defined in Chapter 3. These are Government, Case-marking, and θ-marking. We begin with Government. As noted in (33), the attributes involved are *Gov*, *head*, and *node*. The attribute *node* is independently defined by (36). The core notion of Government, as noted in Chapter 3, is that the *head* of a maximal projection governs exactly those categories that appear within the maximal projection and are not protected by another maximal projection. The attribution that implements the core notion is shown in 3.3.(4) in Chapter 3, and is repeated here in (38).

(38) a. $XB \rightarrow X \quad YP_1 \ldots YP_n$

attribution:

$YP_i.Gov \leftarrow \langle X.node, prodrop \rangle$, for $i = 1, \ldots, n$.

$XB.head \leftarrow X.node$

b. $XP \rightarrow YP_1 \ldots YP_n \quad XB$

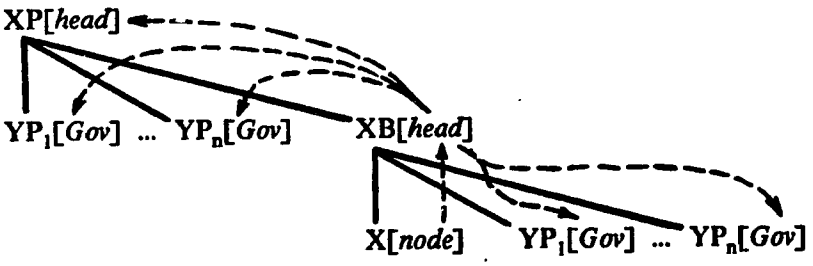
attribution:

$YP_i.Gov \leftarrow \langle XB.head, prodrop \rangle$, for $i = 1, \ldots, n$.

$XP.head \leftarrow XB.head$

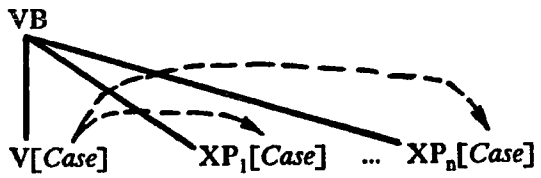
The attribute *head* is synthesized by XB and XP, while *Gov* is inherited by specifiers, adjuncts and complements. We identify the *head* of a phrase by its *node* number, and similarly the *Gov* of a category by the *node* number of the relevant *head*. The attribute *prodrop* is a parameter of the grammar, whose value may be *proper* or *nil* and depends on the category X and the particular language described. If X is V, then *prodrop* has value *proper*, regardless of the language, while if X is I (Inflection), then *prodrop* is *nil* for English and *proper* for Italian and Spanish. The dependency graph that results from the attribution (38) is shown in (39). The dependencies induced in exceptional government are quite similar to those of exceptional Case-marking, as shown next.

(39)

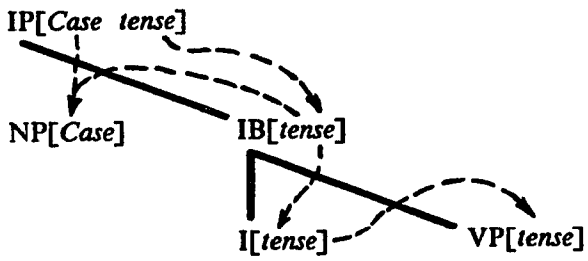


Next we consider Case-marking. From the taxonomy of attributes in (33), the relevant attributes are *Case* and *ecm*. The attribute *tense* is also important in the attribution. *Case* is associated with the categories P, V, NP, and CP, CB, and IP. We will not repeat here the attribution for *Case*, which was derived in Chapter 3. This attribute is inherent in P and V, and its value is determined by lexical properties of the preposition or verbal element inserted under them. The attribute is inherited by the other four categories. The attribute dependencies that arise in Case-marking are shown in the graphs (40).

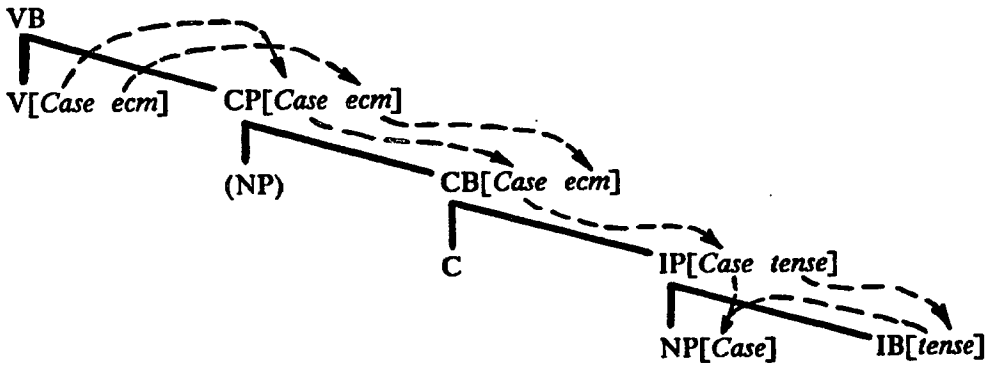
(40) a. Case-marking by V



b. Case-marking of NP in Subject position by [+*tense*]

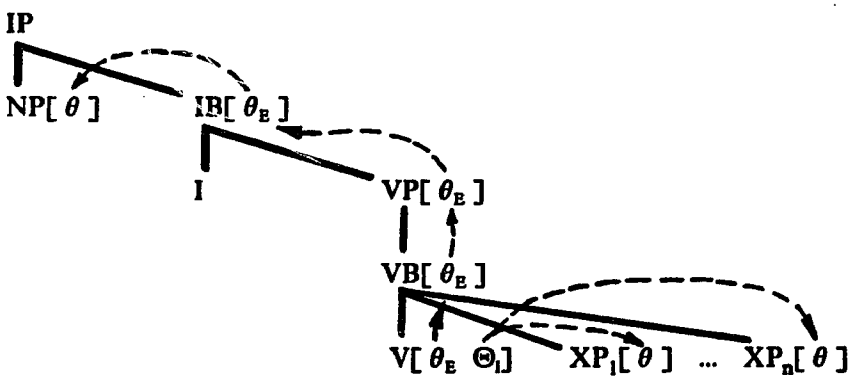


c. Exceptional Case-marking of NP in Subject position



The last item we consider is θ-marking. The attributes involved in this process are θ, θ_E, and Θ_I, as shown in (33). The attribution rules responsible for theta-marking were developed in Chapter 3; the attribute dependencies that arise are shown in (41). Notice that θ-marking of NP in Subject position is mediated by the node IB, and that external θ-marking cannot be evaluated from left-to-right in English.

(41)



As may be infered from the details of the grammar in Chapter 3, we have looked only at a few of the attribute dependencies induced by the grammar. The four cases we have considered, node enumeration, Government, Case-marking, and θ-marking, though, are sufficient for illustrative purposes and the purposes of the following section, on attribute evaluation.

4.4.4 Attribute evaluation

Except for two exceptions, all attribute evaluation assumes only of *unification* of functional terms. Contrary to traditional approaches to attribute evaluation, the unification approach does not require prior evaluation of attributes on which the current attribute depends. The exceptions noted involve testing of attribute conditions (possibly in conditional attribution rules), and evaluation of so-called "computable expressions" in the attribution language. The exceptions require evaluation of input arguments and are accomodated by a limited use of Prolog's bactracking mechanism, assuming a restriction on the domain of the attributes involved. The attribute evaluator we develop is a logic program, like the LL(1) parser (28). The evaluator consists of a collection of Prolog goals G_p, one for each production $p \in P$. The evaluator goals may be applied individually on a tree, and application of a goal G_p has the effect of evaluating the attribution R(p) and B(p) associated with p.

4.4.4.1 Evaluation scheme

Conventional attribute evaluators for attribute grammar operate on a phase separate from parsing and provide no feedback to the latter. In the formulation of the evaluation scheme, it is assumed that a structure tree τ for the input string is available; the evaluation problem then consists in finding a total order on the attribute positions of τ such that each position is data dependent only on positions lower in the order (Deransart and Maluszynski, 1985).

A *static* attribute evaluation scheme determines the evaluation order at construction time from the grammar, independent of the individual trees generated. A *dynamic* evaluator, on the other hand, determines evaluation order depending on the particular input tree. Hence, dynamic evaluation applies to a larger class of grammars. In either case, the evaluator defines a sequence of traversals of the tree which can be used to compute all attribute values and test all attribute conditions. An *incremental* attribute evaluator may compute the attribution in a modified semantic tree without reevaluating the entire tree. The incremental evaluator exhibits optimal time performance if it takes time proportional to the number of attribute instances affected by the modification.

An imperative for analysis with the attribute grammar defined in Chapter 3 is that attribute evaluation be done as a derivation tree is built. This technique is known as *attribute-directed parsing* (Watt, 1980). The techniques of Waite and Goos (1984), Madsen (1980), and Kastens, Hutt, and Zimmermann (1982) for design of attribute evaluators assume that a derivation tree is available before evaluation is done and are non-incremental. Kastens, Hutt, and Zimmermann (1982) consider off-line attribute evaluation without access to the derivation tree, but their method still assumes a parsing stage in which the parser creates an intermediate file with the evaluator actions that would be invoked on the tree being built. Attribute evaluation proceeds off-line when the control file is read and interpreted. In this evaluator, the evaluation order is fixed by the evaluator depending on the grammar, and independently of any input. Madsen (1980) considers dynamic attribute evaluation, in which the evaluation order is dependent on the particular input tree. The parse tree is represented in the form of a right-parse. Yeh (1983) and Reps (1984) describe incremental attribute evaluators for use in syntax-directed editors.

The reason why attribute-directed parsing is needed for parsing with the grammar of Chapter 3 has to do with the incredible degree of ambiguity of the underlying context-free syntax (5). In this context, incremental evaluation would also be extremely helpful. Brief consideration of the

possibilitites for insertion of empty categories shows that the base grammar has a degree of ambiguity which grows exponentially with the length of the input strings allowed: Since CP is a symbol recursive on itself (42.a) and V is the only category that must be overt in that basic CP structure, (5) may generate arbitrarily long sequences in V*. For every element V in such sequences, it is possible to attach or not an NP or PP node, among others (42.b); by the ϵ-productions for the latter two categories, the nodes generated may be expanded into the empty string. Hence, the number of choices for insertion of empty categories in a sequence in V* grows exponentially with the sequence length.

(42) a. $[_{CP} \ldots C\ [_{IP} NP\ [_{IB} I\ [_{VP}\ [_{VB} V\ ]\ CP\ ]]]]$

b. $[_{CP} \ldots C\ [_{IP} NP\ [_{IB} I\ [_{VP}\ [_{VB} V\ (NP)(PP)]\ CP\ ]]]]$

The grammar (5) is limited in the range of constructions it can generate. A minor extension of the grammar (5) to remedy this situation, in the style of Government-binding, is addition of ϵ-productions $V \rightarrow \epsilon$ and $CP \rightarrow \epsilon$. The two productions generate empty categories, which allow for topicalization of clauses and inversion of the copula with a subject. The two productions, though, make the base grammar infinitely ambiguous. The new base defines infinitely many derivation trees for the empty string or any other string.

While transformation of the base grammar (5) to a weakly-equivalent normal form (e.g., Chomsky Normal Form) is possible and would eliminate the problem noted with empty categories,[6] this is an option which we do not consider here since it would require major and non-obvious changes in the attribution. More importantly, we believe it is important to maintain syntactic analyses faithful to those provided by the original Government-binding theory. Such is an original motivation of this work. Most grammatical processes, especially government, make crucial reference to phrase structure; any transformation of the base grammar would require reformulation of most components in the theory.

In spite of the ambiguity of the underlying grammar (5), the phrase-structure it may generate is constrained by certain attributes. This phenomenon is known as *rule splitting* (Watt, 1980). It

---

6 No ϵ-productions would be present in the transformed grammar, assuming the empty string ϵ is not a sentence of English.

may be approached by allowing the behavior of the parser to be influenced by the values of these attributes. The basic approach for parsing is to grow a fully attributed, but partially evaluated derivation tree. The values of the attribute positions which are instantiated may be used to guide the parsing process; that is, for selection of the next production.

The evaluator that we develop is not incremental in the sense of (Yeh, 1983) or (Reps, 1984). The reason is that the LL(1) parsing algorithm we implemented above utilizes the bactracking facility of Prolog to search for valid derivation trees. When backtracking occurs, some of the structure that has already been built is lost. If a different analysis algorithm were used, one that does not discard structure that has already been created, incremental evaluation would be possible and efficiency would be increased considerably. Incremental evaluation is more efficient since it avoids reduplication of work. Only the attribute occurrences affected by a modification of the tree are reevaluated. We believe that the need for incremental attribute evaluation and attribute-directed parsing holds for processing of any natural language grammars that use an attribute-value language as their basic framework.[7]

The attribute evaluator we develop combines attribute evaluation and testing with LL(1) parsing actions. The evaluator is thus attribute directed. The packets R(p) and B(p) of attribution rules and conditions associated with a production p are applied immediately upon application of the production, assuming the top-down parsing model of the LL(1) algorithm in the previous section. This technique has a straightforward implementation in the framework of logic programming, but it may require a restriction on the domains of certain attributes. Furthermore, it does not yield an optimal parsing algorithm. This is discussed in more detail below.

The Prolog parser (28) is augmented as in (43). The unification of the attribute frame *Sym_frame* of the lexical item *Sym* and the frame *B_frame*, excluding the first two positions, of the terminal category B, taking place in clause 2, implements the attribution of the lexical insertion rule (7) above. During expansion of the derivation tree by a application of a production, in clause 3, the subgoals 3 and 4 in the clause compute the attribute frame of each

---

[7] In the PEG grammar of (Jensen, 1986) there is a multi-set of 10 productions VP→NP VP that attach a noun phrase to a verb phrase. The difference between these rules is in the attribution they bear.

category on the right-hand side of the production selected, and apply the attribution associated with that production.

(43) **Attribute-directed LL(1) Parsing**

```
/* 1. Complete parsing of production */
lll_parse( [ Top| StackTail], sit( _, _, []), RestOfInput) ←
    lll_parse( StackTail, Top, RestOfInput).

/* 2. Consume terminal 'B' in input string */
lll_parse( Stack, sit( L, R, [fr( B, B_frame)| RTail]),
                    [ tok( Sym, B, Sym_frame)| RestOfInput]) ←
    append( R, [ fr( B, B_frame)], Rnew]) &
    B_frame = [ node(N), node0(N0)| Sym_frame] &
    lll_parse( Stack, sit( L, Rnew, RTail), RestOfInput).

/* 3. Expand recursively non-terminal B */
lll_parse( Stack, sit( L, R, [fr(B, B_frame)| RTail]),
                  [ Lookahead| RestOfInput]) ←
    lll_production( B, RHS, W) &
    member( Lookahead, W) &
    attribute_frame( RHS, RHS_frame) &
    apply_attribution( B, RHS, B_frame, RHS_frame) &
    append( R, [ fr(B, B_frame)], Rnew]) &
    lll_parse( [ sit( L, Rnew, RTail)| Stack],
               sit( B_frame, [], RHS_frame), [ Lookahead| RestOfInput] ).

/* 4. Termination condition */
lll_parse( [], sit( "Z", _, _ ), [ "#"]).
```

The categories that appear in situations are attributed symbols now. An attributed symbol is a term *fr*(X, A(X)), where X is the category name, and A(X) the corresponding attribute frame. Similarly, the input tokens in clause 2 are terms *tok*(*t*, X, A(*t*)), where *t* is the input symbol, X its category, and A(*t*) the attribute frame produced by the lexical analyzer. This frame differs from A(X) in that it does not have slots for the *node* and $node_0$ attributes.

A goal *attribute_frame*(τ, τ') is provable if τ is a list of categories in the grammar and τ' is the corresponding list, with each category X augmented by its attribute frame, to yield the term *fr*(X, A(X)). The predicate *apply_attribution* is defined by the clause (44), where G_p is a Prolog goal that evaluates the attribution associated with production p. The first two arguments X and α of *apply_attribution* are used to index into the correct G_p. A procedure for constructing G_p, for each $p \in P$, is given below. If evaluation of G_p succeeds, then the recursive call to the *ll1_parse* predicate at the end of clause 3 is made; otherwise, backtracking is forced on the choice of p for the next step in the derivation -- and possibly further back.

(44) apply_attribution(X, α, X_frame, α_frame) $\leftarrow G_p$, where $p = X \rightarrow \alpha$

4.4.4.2 Unification

Let D be a domain, V a set of variables over elements in D, and F a set of functors on Cartesian products of D. Associated with each functor $f \in F$ there is its arity $n \geq 0$. If $n=0$, then f is called a *constant*. A *term* over D is either a constant, a variable, or a functional expression of the form $f(t_1, ..., t_n)$, where f is an n-ary functor and the t_i are terms, for $i = 1, ..., n$. If t is a constant or a functional term with no variables, it is called *ground* or fully instantiated. Otherwise t is called uninstantiated (if t is a variable), or partly instantiated (if t is a functional term).

Let t be a term. A *simple substitution* on t is an operation which replaces all occurrences of a variable in t by another term. A general substitution may make such replacement for more than one variable. If Θ is a substitution, we say that $\Theta(t)$ is an *instance* of t, and t is said to be *more general* than $\Theta(t)$. Θ is a *unifier* of two terms t_1 and t_2 if $\Theta(t_1) = \Theta(t_2)$.

Unification is the basis of most work in automated logical inference and a key operation in the computational models of logic programs (Sterling and Shapiro, 1986). Given two terms t_1 and t_2, unification is an operation that finds the most general common instance of the two terms, if a common instance exists, or reports failure otherwise. We say that t_1 and t_2 are *unifiable* if there is a unifier Θ of t_1 and t_2. A unifier Θ need not exist. In such case we say that the two terms are not unifiable. The reader is refered to Robinson (1979) for more detail on the unification operation and one particular unification algorithm.

Unification may be implemented practically by consideration of the three cases in (45).

(45) Let t_1 and t_2 be terms. Then,

(i) if t_1 and t_2 are the same constant, they unify and their common instance is the constant they represent.

(ii) if one of the terms is a variable, say t_1, and t_2 a term (may also be a variable), the two terms unify and their common instance is t_2.

(iii) if t_1 and t_2 are functional terms $f_1(a_1, ..., a_k)$ and $f_2(b_1, ..., b_m)$, they unify if they have the same main functor ($f_1 = f_2$) and arity ($k = m$), and furthermore a_i unifies with b_i, for $i = 1, ..., k$.

An unification algorithm is typically implemented such that it does not compute the most general instance for the input arguments, but rather a most general unifier (substitution), which yields the common instance when applied to one of the arguments. In the implementation of the algorithms, the assignment of a value to a variable may be done by setting a *reference pointer* from the variable to the value. Thus, in case (ii), unification of t_1 with term t_2 simply sets a reference pointer from t_1 to t_2.

4.4.4.3 Evaluation of attribution functions

Let $p = X_0 \rightarrow X_1 \ldots X_n$ be a production, R(p) the associated attribution rule packet, and r a rule in R(p). An observation crucial for the attribute evaluator we will formulate is that if r is a "copy" rule $X_i.a \leftarrow X_j.b$, which induces a dependency of the attribute occurrence $X_i.a$ on that of $X_j.b$, it may be applied *even if the value of* $X_j.b$ *is not already known*. Suppose v_1 and v_2 are the variables associated with attribute occurrences $X_i.a$ and $X_j.b$. Then, application of the attribution rule simply unifies v_1 and v_2, such that the two variables share. In more loosely stated terms, we unify $X_i.a$ and $X_j.b$. If the value of $X_j.b$ is known when the rule is applied, the value of $X_i.a$ will be instantiated at that time. On the other hand, if the value of $X_j.b$ is not known when the rule applies, the value of $X_i.a$ will remain uninstantiated (but sharing with $X_j.b$). When the value of either variable becomes instantiated to a particular value, so does the value of the other; see for example the dependencies (41) in external θ-role assignment.

If r is a slightly more complicated formula of the form $X_i.a \leftarrow f(X_j.b, ..., X_k.c)$, where f is a functor with $X_j.b, ..., X_k.c$ as arguments, we may deal with evaluation of the rule in exactly the same way as before. Application of the rule creates a binding between $X_i.a$ and the possibly partially instantiated term $f(X_j.b, ..., X_k.c)$. It is not necessary to evaluate the attribute occurrences $X_j.b, ..., X_k.c$ before the rule is applied.

If the expression $f(X_j.b, ..., X_k.c)$ above defines a term via a so-called *computable expression* in the attribution language, the above attribute evaluation method is not readily available. Computable expressions require that the values of all input arguments be known at expression evaluation time; typically they contain arithmetic or string operators. It is necessary to insure that input attribute values are instantiated when they are used in a computable expression. In the attribute grammar we developed computable expressions are used only in the attribution schema (36), which defines the values of the attributes *node* and $node_0$. However, the occurrences of the attribute *node* in the computable expressions of (36) is inherited and data dependent only on attributes of nodes lower in a preorder enumeration of the tree. Given the LL(1) or preorder tree traversal used here for parsing and attribute evaluation, a node's *node*-value is know when the node is visited and its attribution evaluated. The use of the computable expressions in the schema cause no difficulty for the attribute evaluator.[8]

The last and significantly different kind of attribution rule we consider is the conditional rule of the form (46). In the rule, the formula $b(..., X_j.b, ...)$ is truth-valued and its value depends on the value of attribute $X_j.b$. The value assigned to the attribute $X_i.a$ being defined depends ultimately on the outcome of the Boolean test in the condition; the result of the test selects one of the two alternative expressions. As in the case of computable expressions, the Boolean test in (46) cannot be evaluated (in general) unless the value of $X_j.b$ is known. Evaluation of (46) requires instantiation of the value of $X_j.b$.

(46) $\quad X_i.a \leftarrow$ if $b(..., X_j.b, ...)$ then $f_1(..., X_k.c, ...)$ else $f_2(..., X_k.d, ...)$

For certain attribute grammars with conditional expressions, including ours, a solution to the problem in (46) is possible. Note that if the domain $dom(b)$ of the attribute in the Boolean test

---

[8] This single use of computable expressions in the grammar can be eliminated, for example, if we assume a unary representation for the natural numbers.

is finite (i.e., contains finitely many values), it is possible to remove the difficulty by enumerating the values in the domain and using Prolog's backtracking mechanism to select an adequate value, if it exists.[9] For each attribute like $X_j.b$ in (46), we assume a unary predicate *enum_b* which enumerates *dom*(*b*) on backtracking. The attribute evaluator may generate a candidate value *v* for $X_j.b$, apply the attribution rule, and let the parser proceed with the analysis. If the value leads to an incorrect analysis path, backtracking is forced and brings the parser's state to that which existed prior to selection of the candidate value. At this point a new candidate value may be tried. Notice that if the value *v* of the occurrence X.*b* is known when the goal *enum_b*(*v*) is tried, the goal succeeds and the only backtracking involved is that in proving the goal.

The method just described for evaluation of conditional semantic rules may also be applied for testing of attribute conditions.[10] Although that method places a restriction on the domains of certain attributes in the grammar and quickly leads to extremely inefficient attribute evaluators, it offers a transparent and systematic way of obtaining an attribute evaluator for testing the grammar. No restrictions are set on the attribute dependencies induced by the attribution. The efficiency of the evaluator derived depends on the number of attributes for which a generate-and-test approach is actually used to compute their values, and on the cardinality of their domains. In the attribute grammar we are concerned with, the domains of most attributes is binary (cf. the *tense*, *anaphoric*, and *pronominal* attributes). Further, since the assignment of values to attribute occurrences is done in a left-to-right manner, when the occurrences are first encountered, and since backtracking takes place when the first wrong choice is found, not all possibilities in the enumeration are considered by the backtracking algorithm (Knuth, 1974). Hence, the attribute evaluator derived for this grammar may be expected to present acceptable runtimes, and it actually does, as noted in section 4.6. It should be remarked, however, that the worst-case runtime of the evaluator is related exponentially to the length of the input string.

---

[9] The condition just noted on the domain of attributes in Boolean conditions does not apply to attributes whose values are known at condition evaluation time, under the attribute evaluation scheme adopted.

[10] In the Chain rule of Chapter 3 we use the test NP.*Chain*≠0, and a similar one in the associated percolation rules. We rely on having the value of the attribute tested instantiated. Hence enumeration of the *Chain* domain (the integers) is not needed.

4.4.4.4 Attribute evaluator

The procedure we have discussed to derive the attribute evaluator is summarized in (47). Given an attribute grammar G with the restrictions noted above, the procedure yields a collection of evaluation procedures (Prolog goals) G_p, one for each production p. If p is the production chosen for expansion of the derivation tree, the procedure G_p may be applied to update incrementally the attribution in new derivation tree (LL(1) parsing is assumed). If the choice of p in the derivation must be revised (the grammar is not LL(1)), the backtracking operation involved in the process undoes the effects of the application of G_p, and the attribution of the new production selected updates the attribution.

(47) Construction procedure for Attribute Evaluator:[11]

1. Let AG=(G, A, R, B), with G=(N, T, P, Z). For each production $p \in P$, let C(p), T(p), and L(p) be as defined in (48) below.

2. For each production $p = X_0 \rightarrow X_1 \ldots X_n \in P$, define an evaluator goal
   $G_p = E_p \,\&\, R_p \,\&\, B_p$
   corresponding to the attribution of p, as follows:

   2.1 E_p is a Prolog goal which generates candidate values for the attribute occurrences in $C(p) \cup T(p) - L(p)$. For each occurrence $X_i.a \in C(p) \cup T(p) - L(p)$, we require that *dom*($a$) be finite and assume the existence of a unary predicate *enum_a* which enumerates the values in *dom*(a) on backtracting. We let
   $E_p = \&_{X_i.a \in C(p) \cup T(p) - L(p)}$ enum_a(v_i)
   where v_i is the variable associated with $X_i.a$.

   2.2 The goal R_p has the semantic effect of applying the attribution in R(p) (though not necessarily of instantiating all attribute occurrences in AF(p)). We let
   $R_p = \&_{r \in R(p)}$ *translate*(r).
   where *translate*(r) is a Prolog translation of the attribution rule r, given by (49).

---

[11] We use VM/Prolog syntax to define the evaluator.

2.3 Finally, the goal B_p evaluates the attribute conditions in B(p). We let

$$B_p = \&_{b \in B(p)}\ translate(b),$$

where *translate*(b) is a Prolog translation of attribute condition b, given by (49).

(48) Let AG=(G, A, R, B), with G=(N, T, P, Z). For each production $p \in P$,

- C(p) is the set of attribute occurrences of p which appear in a computable expression in some rule or condition in R(p) $\cup$ B(p).

- T(p) is the set of attribute occurrences of p whose value is tested in some rule or condition.

- L(p) is the set of attribute occurrences of p which may be evaluated from left-to-right (Bochmann, 1976), and which depend only on inherited attributes of the root (LHS) of p.

Given the LL(1) parsing model we assume, the set L(p) in (48), for $p \in P$, contains precisely those attribute occurrences of p whose values are known at evaluation time of G_p.[12] Hence, in (47.2.1) it is not necessary to enumerate the attribute domains for occurrences of these attributes, even if they appear in computable expressions or attribute conditions. Notice that the attribute *node* in (36) satisfies this restriction.

The function *translate* in steps 2.2 and 2.3 of (47) may be applied to unconditional attribute expressions, attribute conditions, and attribution rules. In (49.1) we assume that computable expressions in the attribution language use the same syntax of the target language (VM/Prolog). In the absence of a precise syntax for the attribution language we assume, we give the sketch (49) of the *translate* function. Notice that the function is defined for conditional attribution rules, so that the restriction of the function to unconditional attribute expressions is not a shortcoming. The definition in (49.3) for conditional attribution rules does not involke *translate* recursively with conditional expressions. The "simple" attribute expressions in (49.3) are those attribute expressions which are unconditional and not computable expressions.

---

[12] G_p is in turn evaluated when the root of the elementary tree defined by p is visited.

(49) Definition of the *'translate'* function:

1. Unconditional attribute expressions:

   - $translate(t) = t$, if t is a constant or computable expression.
   - $translate(\mathrm{X}.a) = v$, if v is the variable associated with X.a.
   - $translate(f(t_1, ..., t_k)) = f(translate(t_1), ..., translate(t_k))$,
     where f is a functor and t_i an unconditional attribute expression, $i = 1, ..., k$.

2. Attribute conditions:

   - $translate(p(t_1, ..., t_k)) = p(translate(t_1), ..., translate(t_k))$,
     where p is a predicate symbol and t_i an unconditional attribute expression.
   - $translate(\neg b) = \neg translate(b)$, if b is an attribute condition.
   - $translate(b_1 \rho b_2) = translate(b_1)\ \rho\ translate(b_1)$,
     where b_1 and b_2 are conditions and ρ a logical connective ($\wedge$, $\vee$, $\rightarrow$).

3. Attribution rules: Let $r = \mathrm{X}.a \leftarrow t$ be an attribution rule and let v the variable associated with X.a. Then

   - $translate(r) = v = translate(t)$, if t is a "simple" attribute expression.
   - $translate(r) = v := translate(t)$, if t is a computable expression.
   - $translate(r) = translate(b) \rightarrow translate(\mathrm{X}.a \leftarrow t_1);\ translate(\mathrm{X}.a \leftarrow t_2)$,
     if t is the conditional attribute expression "if b then t_1 else t_2."

We have sketched informally a procedure for converting packets R(p) and B(p) of attribution rules and conditions satisfying certain restrictions into a Prolog goal G_p, which may be used to evaluate incrementally the rules and conditions in R(p) and B(p). The goal G_p that (47) yields

for each production p may be optimized in several ways. An item of interest is the relative ordering of the literals in G_p. For example, (47) puts all enumeration subgoals ahead of any attribution rules or conditions. A simple optimization is to reorder the literals in G_p, such that a literal *enum_a(v)* appears as late as possible. It should appear immediately before the first condition or computable expression in which the variable v is used. The procedure (47)-(49) and several optimizations as the one just noted have been applied manually to produce the attribute evaluator used in the parser.

The goal G_p is intended to be executed immediately after application the production p. The evaluator may also be applied on a pass separate from parsing, assuming the parse tree is available. In this case, though, a more traditional attribute evaluation scheme could impose fewer restrictions on the attribution of the grammar. As noted several times before, we assume a preorder tree traversal. A modification of the construction procedure (47) seems possible, to construct evaluators for use in conjunction with LR(k) parsers.

Upon application of the production p to expand the current derivation tree, and before application of the goal G_p, only a subset of the inherited attribute occurrences of the root node of p will be instantiated; all others are associated with uninstantiated variables. The occurrences which are instantiated will be precisely those which may be evaluated from left-to-right. When G_p applies, some other attribute occurrences may become partially instantiated. If the attribute grammar is well-defined and non-circular, all attribute values will be fully instantiated upon completion of the parse tree. Due to the use of unification, the evaluation scheme will work even if the attribute grammar is circular; in such case, the value of an attribute occurrence in a dependency cycle may be a term containing a self-referencial variable (assuming the Prolog implementation omits the so-called *occurs check*, as most do).

4.4.4.5 Discussion

The attribute evaluator we obtain is a simple and practical one, due to the implementation language used and characteristics of the particular attribute grammar we deal with. The evaluator uses crucially unification and backtracking. Unification permits handling in a simple manner "semantic definitions" of the form X.a←t, where t is a term, ignoring whether the arguments on which t depends are instantiated. Backtracking, with the restriction on certain attribute domains, permits evaluation of computable expressions, attribute conditions, and conditional attribution rules by enumerating the values of the domains involved. The efficiency

of the evaluator obtained is related to the size of the sets $G(p)=C(p)\cup T(p)-L(p)$, for each production p. If $G(p)=\phi$, for all p, then the evaluator is deterministic, since no attribute occurrences need be enumerated in (47.2.1). The attribution (36) used in node enumeration, for example, is evaluated deterministically since the condition is met.

The class of grammars U for which the evaluator obtained is deterministic (assuming preorder tree traversal) cannot be compared with the class L of grammars that may be evaluated from left-to-right (Bochmann, 1976). On one hand, the condition on $L(p)$, for each p, that it contain attributes with depend only on inherited attributes of the root, means $U \not\ni L$. Thus, the alternate attribution (50) for node enumeration, which is equivalent to (36) and may be also evaluated from left-to-right, cannot be evaluated (deterministically) by the evaluator defined by (47). The reason is the use of the synthesized attribute $node_0$ in the computable expressions in (50.a).

(50) a. $X_0 \rightarrow X_1\ X_2 \ldots X_n$

attribution:

$X_1.node \leftarrow X_0.node + 1$

$X_{i+1}.node \leftarrow X_i.node_0 + 1$, for $i = 1, \ldots, n\text{-}1$

$X_0.node_0 \leftarrow X_n.node_0$

b. $X \rightarrow \in \mid t$, for t a specified or lexical formative

attribution:

$X.node_0 \leftarrow X.node$

On the other hand, if the attribution in the grammar uses no attribute conditions and each rule is a "semantic definition" of the form (51), where $f(v_1, \ldots, v_n)$ is a functional term in the domain of attribute X.a, the grammar need not be evaluable from left-to-right, but it may be evaluated deterministically by the evaluator produced by (47). We obtain the pure *Functional Attribute Grammars* of (Deransart and Maluszynski, 1985). For these grammars, the evaluator defined by (47) is deterministic, since $C(p)$ and $T(p)$ are empty, for all p, and finds the "basic term interpretation" of the attribute occurrences in each tree.

(51) $X.a \leftarrow f(v_1, \ldots, v_1)$

The attribute grammar we are concerned with, however, cannot be evaluated from left-to-right and uses heavily conditional semantic rules and attribute conditions.

4.5 Extensions of the LL(1) Parsing Algorithm

The parsing automaton developed in section 4.3.4 defines a deterministic parsing algorithm if the grammar from which it is derived has the LL(1) property. The context-free grammar (5) that underlies the attribute grammar, though, is not LL(1), as it has the properties described in section 4.2.2. For this grammar, the automaton is non-deterministic and, yet worse, does not define an algorithm, since it will not terminate for certain inputs. In this section we consider three extensions of the PDA which will ensure termination for all inputs and improve its runtime.

4.5.1 PDA stack length bound

The first extension to the PDA of section 4.3.4 is an absolute bound on the length that the stack may reach during derivation of a string. The objective of this constraint is to ensure termination of the parser during analysis of any input string. The constraint alone is not sufficient to ensure termination for an arbitrary CFG.[13] However, the grammar (5) is an instance of an X-bar grammar (Kornai, 1983); given the stack constraint, the X-bar property is sufficient to ensure termination for an arbitrary input.

The relevant observation about X-bar grammars that we need to prove this claim is that the productions satisfy the schema (52), where $0 < n \leq max_X$ and the XP_i are maximal projections.

(52) $\quad X^n \rightarrow (XP_1) \ldots XP^{n-1} \ldots (XP_k)$

Assuming (52), recursion on a given category XP is possible in the course of a derivation only through an intermediate maximal projection YP. That is, if XP is on top of the LL(1) stack and the same category appears in that position after a number ($\geq$1) of moves, then there is a maximal projection YP which appeared on top of the stack after the first appearence of XP. It

---

[13] Let G be a grammar with the production set {Z → X, X → *a*, X → X }. Then there is an infinite derivation Z ⇒ X ⇒ ... ⇒ X ⇒ ... with a stack of length 1.

is possible that YP=XP. But the maximal projection YP is introduced only as a sister of some head X^k. If YP preceeds X^k, the length of the stack will grow by at least one for each recursion cycle, since the associated heads X^k will accumulate on the stack. If YP follows the head, the size of the stack need not grow, but then at least one input symbol is consumed from the input string, as follows. The head X^k eventually leads to the zero-level category X^0. If X^0 is a lexical category, it consumes an input symbol.[14] If X^0 is non-lexical (C or I) a symbol need not be consumed right then, but then YP = IP or CP, and we can show from (5) that a verbal element is consumed. In either case we must conclude that the depth of recursion is bound, by the stack length bound, by the length of the input string, or by a combination of both. Hence the PDA terminates on all inputs.

A first approach to the PDA stack length bound is to relate it linearly to the length of the input string. Some experimentation and psycholinguistic observation, however, show that a fixed bound is appropriate. First, typical constructions in English are right-branching, with small degree of center embedding, as in (53). During derivation of these constructions, the size of the stack remains relatively stable at a low number. For example, the derivation of (53) need never grow the stack beyond three cells.

(53) I think that Bill said that John believes that ... Mary came.

In cases where the length of the stack grows steadily in the course of a derivation, as in center-embedded constructions (54) or left-branching constructions (55), there is a sharp limit in the degree of embedding that is acceptable. Notice that all these examples are grammatical in the sense that they are generated by the grammar, but that they differ widely in acceptability. While the constructions in (a) and (b) are clearly acceptable, those in (c) and (d) are less so. A fixed stack length bound constrains the depth of recursion allowed in these derivations and limits acceptance of the strings, mirroring the human performance constraint.

(54) a. The dog jumped.

b. The dog that the cat chased jumped.

---

[14] This is true of the grammar (5). However, in the Government-binding framework lexical heads can actually be empty -- but a terminal item associated with the head must appear elsewhere in the tree.

c. ? The dog that the cat the mouse catched chased jumped.

d. ?? The dog that the cat the mouse I saw catched chased jumped.

(55) a. John's car

b. John's car's engine

c. ? John's car's engine's carburator

d. ?? John's car's engine's carburator's valve

Hence, for English it seems possible to use a stack bound which is fixed, say ℓ_{max}, independent of input length, as in (56). In a language like English, and assuming a PDA which traces derivations top-down, the fixed stack bound is a performance constraint to which there is a high degree of plausability.

(56) **Stack length constraint:**
$*\ [X_1\ X_2\ \ldots\ X_n]$, if $n > \ell_{max}$

Adoption of a stack length bound changes significantly the language recognized by the PDA from that generated by the grammar. The effect of the fixed stack bound on the language recognized by the PDA must be viewed in the context of the branching properties of the language generated by the grammar and the parsing model implemented by the automaton. If the language is left-branching (e.g., Japanese), a top-down PDA is likely to grow its stack proportionately to the length of the input string. If the language is right-branching but the PDA is bottom-up, derivation of the same construction (53) will grow the stack proportionately to the length of the input.

4.5.2 Illegal PDA stack configurations

Certain PDA stack configurations may be declared illegal in the parser, in order to improve performance. A family of configurations is declared illegal by making reference to the top k positions of the stack, for some fixed $k \geq 0$. The effect of the declaration is to block all moves of the automaton when the top k symbols of the stack match the declared pattern. In this case bactracking is forced and the parser must consider some alternate derivation path, if it exists.

The only PDA stack configurations which have been declared illegal in the parser implemented are those that match the pattern (57), where x and y are variables that range over the stack alphabet Γ and z over Γ^*.

(57) **illegal stack configuration:**
$*$ [$x, y, y, y \mid z$]

Among the stack configurations ruled-out by (57) are those in (58). (58.a) is encountered in the derivation of (54) above, while (58.b) in the derivation of sentences like (55). The constraint (57) thus overlaps with the stack length constraint (56), in that both constrain derivations with arbitrary amounts of center embedding or left-branching.

(58) a. [N, IB, IB, IB | z]
b. [NP, NB, NB, NB | z]

4.5.3 Feature lookahead

The LL(1) predictor of the automaton (25)-(26) in section 4.3.4 makes reference only to the syntactic category of the next input token and the category of the top symbol of the stack in order to select a production to apply. This feature of the parser stems from the method used to define the predictor, which ignores the attributes associated with categories in the grammar. The director sets computed by the LL(1) parser generator contain only syntactic categories, and the predictor effectively ignores all features associated with the two categories referenced.

In certain situations, though, the accuracy of the predictor can be sharpened considerably by giving it access to the syntactic features associated with the next input symbol and the top symbol of the stack. This elaboration of the predictor may be seen as an extension of the

attribute-directed parsing discussed in section 4.4. Consider, for example, the configuration (59), in which the parser is ready to parse a VB while the next input symbol is *like*.

(59)	**Stack configuration**	**Rest of input**
	[VB| *z*]	*like* ω

Quick inspection of the productions in (5) reveals that there are nine options for selecting a production when the parser configuration is (59); given that the predictor uses only the category of the next input symbol, it is unable to reduce the number of options available. By making available the *subcat* attribute of the next input token, if it is a verb, it is possible to reduce the options of the predictor to just one, namely the production VB → V XP, where XP is the category in the subcategorization frame of *like*.[15]

The implementation of one-symbol feature lookahead has been done manually and only for a few productive cases, like the verb subcategorization frame just described. It is possible to include the attribute frame associated with a category, or a subset thereof, as part of the definition of the category. This is done in GPSG (Gazdar et al, 1985); the category V is split into nine subcategories, according to the nine subcategorization frames made available by the productions. These nine subcategories could be split further, taking into account the other attributes associated with V, such as external θ-role, etc. Since the attribute frames associated with syntactic categories are fixed, and the domains of attribute values are not recursively defined and are in most cases finite, the PDA construction procedure could be extended so that it applies to complex (attributed) grammatical symbols. In this case, the director sets produced by the LL(1) parser generator would consist of complex symbols, and feature lookahead would be built-in directly into the LL(1) predictor.

The drawback of this approach is clearly that the number of categories (and hence productions) in the grammar is increased exponentially, according to the cardinality of the domains of attributes associated with the original atomic categories. The cardinality of the director set associated with each production may also increase exponentially, placing greater demand on storage requirements and computation time needed to perform set operations. A tradeoff has been made by selecting a small number of attributes for lookahead and by implementing the necessary changes to the predictor manually.

---

[15] The verb *like* may subcategorize for a noun phrase NP or clausal complement CP. This kind of lexical ambiguity is accounted for by two different lexical entries for the verb and occurs at the lexical lookup stage.

Feature lookahead has also been implemented for other attributes in the grammar, such as *Case* inflection on nouns and *tense* on verbs. This kind of lookahead eliminates a lot of unnecessary bactracking during parsing.

4.6 Parser Performance

We have provided extensive detail regarding the parsing program which has been implemented for the attribute grammar of Chapter 3. The procedure that was followed to obtain the parser from the grammar is fairly systematic, and we believe could be fully automated. It is general enough to apply to a significant class of attribute grammars. Now, we limit our comments about performance to the simple observation that its worst-case behavior is exponentially related to the length of the input string. This behavior stems from two facts about the grammar: (i) The lack of the LL(1) property of the underlying CFG, and (ii) the use of a generate-and-test method for evaluating the values of some of the attributes used. These observations may be illustrated with two examples.

Consider first the string (60.a) and the syntactic structure associated with it. In this structure there is the possibility of attaching a PP as in (60.b) at at least three places (namely as a complement of *job*, a modifier of *get*, or a modifier of *tell*). The parser generates each of these analyses successively, on backtracking. Now consider the case where there are two PPs, as in (60.c). There are two attachment possibilities for the first PP, and either one or two for the second, depending on where the first is attached. As the number of embedded clauses in the matrix sentence increases, it is clear that the number of possibilities that need to be considered increases exponentially.

(60) a. Nancy told Mary that Bill got a job.

b. Nancy told Mary that Bill got a job in Cambridge.

c. Nancy told Mary that Bill got a job in Cambridge in NYC.

The second case of backtracking can be shown in an even more straightforward manner. From the attribution rules associated with the production for IP, it is clear that NP.θ is one of the attributes for which a generate-and-test method is used. The parser assumes by default that an

external theta-role is assigned by verbs. That is, we assume first IB.$\theta_E \neq nil$. This assumption is verified with the value of the attribute θ_E associated with V, when the main verb of the clause is found. Where this is not the case, backtracking is forced. In parsing (61.a) no backtracking is involved (due to this factor), while in (61.b) one backtrack operation is required during parsing of *seem* and in (61.c) the amount of backtracking involved (on the choice of IB.θ_E values) is linearly related to the length of the input.

(61) a. John likes Mary.

b. It may seem that John likes Mary.

c. It may seem that ... it seems that John likes Mary.

We have not considered the effect of PDA stack constraints on parser performance. It is clear that the constraints improve performance, at the expense of altering the language recognized by the automaton.

Several sample runs of the parser for typical inputs were obtained using a version of the parser implemented in VM/Prolog, running on an IBM 3090 computer. The results of these runs are shown in the Appendix. The runs show the handling of movement structures by the parser. The inference speed of the system was measured to be 90,000 logical inferences per second (90K LIPS). The parsing time for the typical sentences shown is between 100 and 400 milliseconds. The sample runs show that the parser performance, as a prototype, is acceptable for typical inputs.

Chapter 5. Discussion and Conclusions

This thesis has presented an interpretive version of the Government-binding theory and defined it explicitly in the formalism of attribute grammar. The main element of the new theory is an interpretive chain formation rule (*Chain rule*), with similar empirical effects to the transformation move-α in Government-binding. Attribute grammar has proved to be a framework suitable for the study of natural language. The formalism raises concrete questions of realization and permits their discussion in a precise manner. One motivation we had for adopting attribute grammar was the availability of well-developed and understood methods to derive analysis procedures from a given attribute grammar definition. This permits the study of performance for Government-binding grammars.

We now wish to discuss some implications of the work presented and suggest some directions for future work.

5.1 Elimination of transformations

From a linguistic viewpoint, the main result in this thesis is the Chain rule of Chapter 3. This rule shows that the transformational formalism used in earlier work for the description of natural language is not the only alternative to phrase structure grammar. Instead, the empirical effects of transformations may be reduced to attributed definitions of the sort developed here. In any case, transformations play only a reduced role in GB grammar. Move-α and the trace convention define only a small number of the properties associated with chains. Most properties follow instead from other components in the theory, especially Case and thematic theory.

The shift from transformational grammar to attribute grammar reduces the expressive power available to a linguistic theory. However, we claim, this reduction does not entail a loss of descriptive power. This is in principle, given the Type-0 generative power of attribute grammar,

and in practice, given the close empirical equivalence of the Chain rule and move-α. We believe that the reduction does not entail either a loss of explanatory power -- i.e., lack of universality of the interpretive mechanisms that may be formulated. Even though the rules we have proposed here are tied to specific phrase structure rules of English, they are so tied in a schematic fashion. For example, the order of constituents on the right-hand side of a production does not play a role in the Chain rule. To the extent that the substantive properties of move-α and chains are universal, the Chain rule may be argued to be universal. We believe that interpretive mechanisms can be embedded in a parameterized (attribute) grammar from which the grammars of particular languages may be derived, by instantiation of its parameters. Development of a parameterized grammar is in fact one of the directions for future work we will suggest.

An important construct of GB grammar is the notion of *empty category*. This notion is fully adopted here and is crucial in the statement of the chain rule. Empty categories may be base-generated and justified independently in a non-transformational theory of grammar. Explicit representation of predicate-argument relations is perhaps the chief motivation. We believe that the typology on empty categories discussed in Chapter 2 is also an important element in the theory of grammar. In Chapter 2 we saw that the typology may be defined without reference to antecedents, at any level of representation at which government, Case, and thematic relations are identified. The determination of empty categories may be carried out independently of the operation of transformations and the manner in which the categories arise. Empty categories and the typology on them may be maintained in a grammatical model in which *all* empty categories arise by operation of the base.

An important question that must be answered by an interpretive account of movement is whether it can indeed explain all grammatical phenomena without reference to transformational *derivations* or, equivalently, whether the phenomena may be described by access to phrase structure representations exclusively. Note that annotated derivation trees abstract away from several details of derivations, including order of application of rewriting and attribution rules. In early transformational grammar, derivational history plays an important role in the definition of grammaticality. This is seen clearly in Emonds' typology of transformations and the elaborate mechanism of the transformational cycle. The role of derivational history in GB grammar is greatly reduced compared to earlier transformational grammar. However, this is a complex problem which we now briefly discuss.

Recent work by Lasnik and Saito (1984) suggests that derivational history may be needed in an account of grammaticality. Their formulation of move-α leaves a trace at the extraction site optionally, rather that obligatorily. This entails that Subjacency must be taken as a condition on derivations, rather than on a level of representation, since it does not seem possible to determine from a single S-Structure or LF representation whether two phrases which have been coindexed by application of move-α were coindexed by cyclic application of the rule, satisfying Subjacency. The truth of this observation, however, depends on what is understood by S-Structure and LF. If what is understood is an un-annotated syntactic tree, then the observation is plausible.[1]

If we understand the annotated syntactic tree, however, the observation does not hold for attributed theories. In attribute grammar, long distance relations between nodes in a derivation tree reduce to relations between adjacent nodes in the tree. In the Chain rule of Chapter 3, for example, the relation between a trace and its antecedent is carried by the ancillary attributes *A-Chain* and *AB-Chain*, present in nodes on the path between the trace and the antecedent. The traces in C-specifier position are required by the Chain rule only for "decorative" purposes, in the following sense: in a correctly attributed tree, C-specifier position is present if and only if the value of the *AB-Chain* attribute at the corresponding CB node is non-zero. Thus, it is possible to infer the presence of a trace from the value of the *AB-Chain* attribute, and it is possible to determine Subjacency by reference to attribution in the tree, even if traces in C-specifier position are omitted; the relevant information is present in the *A-Chain* and *AB-Chain* attributes.

It is difficult to determine whether all restrictions that the transformational cycle imposes on representations derived by transformation may be expressed as conditions on the representations themselves. Some of these conditions can indeed be expressed. The Chain rule characterizes syntactic chains which obey the Subjacency condition while also observing the principles of the transformational cycle. See for example the remarks on the examples (22) in Chapter 3. The development of transformational grammar has been away from conditions on derivations (or rule application) to conditions on representations (or the output of the rules themselves) (Chomsky, 1981). This theoretical direction is highly compatible with the requirements given by the attribute grammar model.

---

[1] Although, under fair assumptions, if the S-Structure tree is not annotated *Chain* indices are not present and the question of Subjacency does not arise, since chains are not identified.

Independent of the above considerations, a reason for concern in the direction of current Government-binding theory is that it assigns increasingly complex representations, and even more complex derivation sequences, to what appear to be uncomplicated strings. An example of this direction is found in (Lasnik and Saito, 1984), who multiply the number of options open to grammatical rules. The rule in question is move-α. It is proposed that the landing site for each application may be base-generated or adjoined, while the trace at the extraction site is left optionally, not obligatorily. Clearly, under this conception move-α is not a grammatical transformation, but a *family* of four transformations, each with different structural condition and structural change. Note that the standard formalism of transformations (e.g., Chomsky, 1957) does not assume disjunction or optionality in the statement of the structural change of a transformation. Once a factorization of an input phrase-marker is determined, the output-phrase marker is uniquely determined.

While the complexity of the representations assigned and the derivation sequences required is ultimately an empirical issue, we believe that a shift toward a framework of attributed definitions will pose questions of theoretical interest concerning economy in the grammar: This is a multi-faceted question which includes size of the grammar, regularity of its components, both for a given language and across languages, and also economy of the representations it produces and the length of the derivation sequences it defines. It was considerations of this sort that motivated (Ross, 1967a) to propose operations such as tree pruning, whose objective was to reduce the complexity of the representations produced.

We close this section with some remarks on quantifier raising. Although quantifier raising was not implemented in the attribute grammar, we wish to discuss briefly how an interpretive account may be given. The empirical effect of quantifier raising is definition of operator scope. A direct implementation would assume tree transduction between S-Structure and Logical Form. This transduction is not problematic for an analysis procedure, as move-α is, since it is not involved in the derivation of the surface string. Quantifier raising may also be defined interpretively, operating on extended S-Structures, as suggested by Aoun and Epstein (1988). A scope can be assigned to each operator in an input S-Structure, by inserting and identifying the node in the structure that corresponds to the LF landing site of the operator in the transformational account. Aoun and Epstein (1988) use scope assignments of the form $<$*Quantifier, landing-site*$>$, in which the scope of the quantifier is the c-command domain of the landing site.

An alternative of interest is the definition of operator scope by domination rather than c-command. Since landing sites are formed by Chomsky-adjunction to IP and not present at S-Structure it may be more appropriate to define scope by identifying the IP node which *dominates* the operator scope -- i.e., the node to which the Chomsky-adjunction would be performed. Note that under the analysis of (May, 1985), all operators adjoined to the same IP projection commute. Hence the different orderings of operators in a representation that may be induced by different sequences of application of quantifier raising are superfluous. Under this convention, explicit representation of the landing sites in a tree is superfluous, in the same sense that intermediate traces in C-specifier position are superfluous. This approach to operator scope we envision is parallel to the way in which the binding domain of an anaphor or pronominal is defined by domination, by its Minimal Governing Category. As in the case of the Chain rule, the approach will be feasible, just in case access to LF derivations is not required to define the notion of grammaticality. If this is the case, two interpretive rules Quantifier scope and the Binding derive Logical Form in a simple manner, by further annotation of S-Structure.

5.2 Locality and modularity of grammatical rules

In the Government-binding model that we adopted as a point of departure (Chomsky, 1981), complex grammatical processes are explained in terms of simple and general principles and rules located in the various components of the theory. In contrast, in the Standard Theory the same processes are given unitary account by means of a single rewriting rule or condition. An example of this trend is the change in the account of the passive construction. In the Standard Theory (Chomsky, 1965), the passive is derived by a single transformation. In Government-binding it receives a more elegant account, as the result of interaction between move-α, Case, and thematic theory. The larger phenomena of passives is descriptive and results from the interaction of simpler elements.

In the attribute grammar of Chapter 3, the principles and rules of Government-binding grammar are not mapped into a single attribution rule or condition, but rather have to be associated with larger fragments of the grammar; that is, with collections of attribution rules and conditions, often involving several productions. The operation of the GB principles and observed surface phenomena may be understood only by considering the behavior of the attribution in the grammar as a collective system. Because of this, the grammatical principles of GB grammar

should be seen as high-level specifications or abstractions over the attribution statements. We return to this point in the following section.

Although it is hard to expose the modularity of an attribute grammar language definition, since the underlying phrase-structure grammar always creeps close, it is clear that all grammatical processes (e.g., building a chain, or correctly assigning a referent to a pronoun) ultimately reduce to extremely simple and primitive attribution operations. These operations are often part of the explanation/implementation of several more complex and readily observable grammatical phenomena. For example, the test on Subject θ-role made in the attribution of the Chain rule, in the production that expands the IP symbol (Chapter 3, 20.c), is at once part of the mechanism of chain formation and contributes to the θ-Criterion. Hence, one of the principal motivations for modularity, namely, economy and decomposition of a definition into more primitive elements, is a property of attributed definitions.

The attribution statements in an attribute grammar are given on a production by production basis. Each attribution statement has access only to the attribute occurrences of nodes in one elementary tree. For example, the relations that the Chain and Binding rules define always reduce to relations between adjacent nodes in derivation trees. In the Chain rule, corresponding to application of each production the Chain rule refers to several attributes of nodes in the elementary tree, including government, Case, and thematic role. Somewhat surprisingly, the well-formedness conditions on chains follow from the operation of the rule and need not be stipulated as filters on the output of the rule. These conditions include Subjacency, the binding conditions, the Case filter, and the thematic criterion.[2]

Similarly, the Binding rule of section 3.6 in Chapter 3 derives LF representations that satisfy the Binding axioms, without employing the filtering scheme of (Chomsky, 1980), and standard in Government-binding. Evaluation of a binding axiom at a given node requires access to non-adjacent nodes in the tree, and in general to the complete tree. The Binding rule does not encode directly the Binding axioms or the notion of local domain; the axioms and notion rather follow from the way in which NP references are added to and removed from the sets of potential anaphoric and pronominal antecedents (*AAS* and *PAS*), at various nodes in S-Structures. The

---

[2] This result is surprising since nowhere in the formulation of the Chain rule are chain conditions directly encoded. Instead, the reference the rule makes to properties of nodes (Case, θ-role, etc) to determine when to start, terminate, or assign a trace to a chain, are sufficient to imply the overall chain conditions.

computation of the *AAS* and *PAS* sets is defined on a production by production basis, without requiring access to a complete S-Structure tree at a time. This feature sets our scheme apart from other approaches (Hobbs, 1978) which require access to complete syntactic trees.

5.3 Abstract specifications and procedural realizations

A distinguishing feature of Government-binding grammar has been its move from rule-based descriptions of language to axiomatic or *principle-based* descriptions. Constrained rule systems that generate only acceptable sentential forms over a given vocabulary have been replaced by extremely general generative rules. Due to unwanted application and interaction, the rules may generate many spurious structures and annotations and must be supplemented with principles or conditions on the structures that may be accepted. In attribute grammar terms, this is a move from grammars in which the underlying context-free grammar defines most of the syntax of the language, by means of a production set that tightly characterizes the syntax, to grammars in which most syntactic definitions are deferred to attribution rules and conditions, while the production set overgenerates. This approach is taken one step further in Government-binding, where no effective function is given to compute some attribute values. Instead, possible values are defined by a set of constraints on them.

Specific examples of the move from constrained rule systems to overgeneral ones include the Case filter (Chomsky, 1981), a filter on distribution of lexical noun-phrases and variables, the Subjacency condition (Chomsky, 1977), a filter on the output of the transformational rule move-α, and the Binding axioms (Chomsky, 1980), which are constraints on referential indices. The efforts of Koster (1978) and Barss (1983) to derive trace-antecedent relations interpretively (i.e., without recourse to transformations), are also *principle-based*, in the sense noted above. In Koster's work, the conditions proposed on the relations are stated in the format of *filters*, advanced in (Chomsky and Lasnik, 1977). Similarly, the characterization of non-argument chains in Barss' work assumes a linking rule that applies freely, and whose output is subject to several output conditions.

The research program towards a principle-based grammar is motivated by the desire to extract only the most general and universal elements from the formulation of language-specific rules. Principles on derived structure compensate for the overgeneration that unwanted rule application and interaction produces. It is implicitly assumed that simpler rules and principles have a greater

change of being universal, while their interaction accounts for the actual grammatical phenomena that is observed. An attractive by-product of the principle-based approach is that axiomatic characterization of grammatical structures is often more succinct than a rule-based grammar, with an elaborate set of rules that generate without filtering exactly the structures intended. The grammatical statement may also achieve maximal simplicity and symmetry (van Riemsdijk and Williams, 1986).

A drawback of the principle-based approach is that it widens the gap of explanation between the theories of competence and performance. While the situation seems to improve for the theory of competence, it gets worse for that of performance. This drawback must overcome in an explanatory linguistic theory. The theory must explain the manner in which the descriptive devices it assumes are put to use in language generation and understanding. Where no non-trivial interpretation of the devices is found,[3] an equivalent more procedural model of the grammar can serve as the basis for a theory of performance and may contribute further insight into linguistic phenomena. It is a difficult problem to find and justify a procedural model for a principle-based grammar. This problem has been noted in recent research which attempts to provide computational models of language for Government-binding (Berwick and Weinberg, 1984; Barton, 1984; Abney and Cole, 1986; Correa, 1987, 1988; Johnson, 1987; Epstein, 1988).

On this issue, Berwick and Weinberg (1984) remark that "the theory of computation forges a vital link between *what* and *how*." They add the notion of *type transparency*, to address the relation between the rules and principles of the grammar with the computational model. The type transparency of the Marcus parser used by Berwick and Weinberg to provide a computational model for Government-binding is not high; they remark that "the Marcus parser does not mention any of these [GB] subtheories explicitly." Instead, the Marcus parser precomputes the effects of the Government-binding principles.

We believe that the gap between the competence grammar and the Marcus parser is due to two main factors. One is the principle-based approach of the grammar; the second is the low-level nature of the operations required in the Marcus parser. This gap is reduced by moving away from either of the two extremes. We believe the type transparency of an attribute grammar definition of language improves over that of the Marcus automaton, due to the higher level of

---

[3] A non-deterministic generate-and-test interpretation is always possible.

abstraction it assumes over that automaton or any other automaton. As we saw in Chapter 3, attribute grammar is neutral regarding the automata that may be used for parsing, and to a large extent also neutral about the attribute evaluation scheme. On the other hand, attributed definitions as developed here are far more explicit and procedural than most principles of Government-binding.

Contrary to what is implied by the motivations for the principle-based approach, it is not clear that constrained generative rule schema need be language specific and with no chance of universal validity. It is often true, though, that rule-based grammars are less succinct than their principle-based counterparts. Furthermore, rule-based grammars contain a lot of detail which, while highly relevant for models of performance, and with a high degree of psychological plausibility, is probably irrelevant at higher levels of abstraction.

To illustrate, we believe that the Chain rule developed here, assuming a presumably universal typology of empty categories and universal subtheories of Government and Case, may turn out to be universal, if we allow factoring out the order of the symbols on the right-hand side of the productions on which it is defined. A similar statement may be true about the procedural Binding rule described in Chapter 3 (Correa, 1988). We do not fully advance the claim about the universality of the rules noted since we have not explored the formulation of the same rules for languages other than English. As anticipated, the rules incorporate certain amount of detail and ancillary attributes, like *A-Chain* and *AB-Chain* for the Chain rule, an *AAS* and *PAS* for the Binding rule, which are probably not relevant elsewhere in the grammar.

We believe that the relation between the principle-based approaches to grammar and the performance grammars that may actually be employed by native speakers in language production and understanding is that of a high-level or *abstract specification* to a program that meets the specification. The distinction between a specification and possible refinements of it is rather well established in computer science and software development (Jones, 1980). While abstract specifications are usually stated in a declarative language, like first-order logic or set theory, with a great deal of expressive power, particular realizations are given in a language with procedural interpretation. The problem of refining a given specification into an realization with optimal performance is a difficult one, which typically must be done manually and involves substantial creative work. The approach we have taken to arrive at the Chain and Binding rules is of this kind.

If a procedural interpretation is available for a specification language, then it is in principle possible to treat specifications as executable programs. This interpretation is of practical interest, however, only in limited and trivial cases. A high level language like Prolog (Clocksin and Mellish, 1978) permits writing of declarative programs, close to an original specification. However, the writing of practical programs often requires careful consideration of the procedural interpretation associated with the language. This introduces implementation bias and detracts from the goal of abstract specifications. Automatic program synthesis from a given specification is still a distant goal, for which there are no general procedures that may be guaranteed to always produce an acceptable program.

As an example of the kind of transformations that may be used in program synthesis, Tamaki and Sato (1984) propose application of the fold/unfold program transformation to logic programs, to derive more efficient programs. Johnson (1987) considers the problem of derivation of performance grammars for Government-binding from a principle-based competence specification. His approach assumes formal manipulation of the competence grammar, by application of the fold/unfold transformation, to derive the performance grammar. Theorem-proving techniques are then applied for direct parsing with the transformed grammars.

The point we have tried to convey in this section may be illustrated best by means of an example. Consider a binary relation *sorted* that holds between two lists of numbers A and B if B is a "sorted" instance of A. This relation may be easily defined axiomatically by two predicates *is_permutation* and *is_ordered*, as in (1). The predicate *is_permutation* holds of two lists A and B if B is a permutation of A, while the predicate *is_sorted* is true of its argument if it is an ordered list, according to some predefined order. One possible way to define these predicates may be found in (Clocksin and Mellish, 1978); we omit them for brevity.

(1) sorted(A, B) ← is_permutation(A, B) & is_ordered(B).

While the specification (1) may receive direct procedural interpretation, for example, the one that a Prolog interpreter gives to it, it is clear that direct use of the specification as a procedure for sorting lists of numbers is extremely inefficient, and in fact completely impractical for lists consisting of more than a few numbers. The running time of this procedure is O(*n*!), where *n* is the length of the input list. While the definition (1) of the relation is maximally concise and simple, it would be erroneous to assume that the noted procedural interpretation of (1) is the best model for how numbers may be sorted. The fold/unfold transformation of Tamaki and Sato

may be used to transform the sort-by-permutation program to a program with runtime complexity $O(n^3)$. They remark that further extension of their method yields an $O(n^2)$ program. Manual refinement of the sort-by-permutation specification may yield in the hands of a creative individual the Quicksort algorithm (Hoare, 1962), whose complexity is $O(n*\log(n))$. The top-level clauses of a Prolog coding of this algorithm is given in (2); the algorithm may be shown to be far superior to the abstract specification (1), although it is logically equivalent to it.

(2)
```
sorted( [], []).

sorted( [H| T], B) ←
    split( H, T, A1, A2)  &
    sorted( A1, A1s)  &  sorted( A2, A2s)  & append( A1s, [H| A2s], B).
```

The distinction we have noted between grammar as specification and grammar as a model of actual linguistic competence and performance raises some interesting questions about the status of the grammars formulated in most current linguistic theory: In what sense can these grammars be said to be innate or constitute the knowledge of language that a speaker has? Given the abstractness of principle-based grammars and their remoteness from actual models of linguistic processing, it seems that principle-based grammars are best seen as *specifications* of the linguistic knowledge that a speaker has, rather than as actual instances of this knowledge. Notice that in this context the *form* of (the formula expressing) the knowledge carries as much weight as the (propositional) content it has.

5.4 The parser implementation

The parsing program for English developed from the attribute grammar, as detailed in Chapter 4, consists of about 4,700 lines of program code and is written entirely in Prolog. About 1,200 lines contain a hand-coded version of the attribution rules and conditions in the grammar, 800 the parser and related routines, 1,500 the lexicon, and 1,100 the LL(1) director set generator. The lexicon consists of 400 lexical entries, with some simple morphological routines to derive inflected forms of verbs and nouns from their lemma forms. The attribute component was hand-coded according to the construction procedure of section 4.4 in Chapter 4 for the attribute evaluator.

The parser implemented offers good practical performance due to the extensions noted in Chapter 4 to the basic LL(1) algorithm. The parser also offers good coverage of typical sentences found in linguistics textbooks. However, the parser is not broad coverage for English. It needs significant extension to handle a wider range of constructions and in its vocabulary. The constructions which seem most needed are topicalization, clefting, coordination, punctuation, and noun compounding. The first two, topicalization and clefting, can be accounted for straightforwardly, by an extension of the Chain rule, as noted in Chapter 3. This extension of the rule is not used in the parser's grammar. The last three constructions seem more parochial, but they, particularly coordination and noun compounding, are absolutely crucial for analysis of actual linguistic text. The constructions noted are quite difficult to integrate into a grammar due to one major problem they multiply, ambiguity.

Extension of the range of constructions and vocabulary of the grammar used by the parser presents one major technical and practical difficulty: the degree of ambiguity of the grammar increases exponentially with the vocabulary and combinatorial possibilities of the grammar. The grammars provided by current linguistic theory are often capable of providing the correct structural description for the most likely interpretation of a given input string. What is often not realized in theoretical circles, most likely due to lack of computational tools to test grammatical theories, is that a natural language grammar often provides many alternate structural descriptions for the same string. Although plausible, native speakers are very rarely aware of all the possibilities. A mechanical analysis procedure, though, computes all the alternatives by design. Thus, it seems that another dimension that should be added to natural language grammars is a *ranking* of the alternative descriptions made available for the same string. Heidorn (1976) proposes one such metric, assuming in part Kimball's (1972) principles of surface structure parsing.

The point of actually programming and testing the parser was to show the technical feasibility of deriving a practical analysis program from the attribute grammar definition. Progress in linguistic theory, and more advanced methods for parser construction are available to derive more efficient parsers and with better coverage.

5.5 Directions for future work

Government-binding theory is a complex linguistic theory in a state of development. We believe that the most accurate characterization of the attribute grammar proposed here is that it is a partial implementation of one of several alternatives made available in the main source of the theory (Chomsky, 1981). Some elements from more recent literature, however, were added (Chomsky, 1982; van Riemsdijk and Williams, 1986; Chomsky, 1986b). An obvious direction in which the present work should be pursued is extension and updating of the theory that is implemented here. Some items that should be considered in the extension are a more detailed account of the ECP, implementation of Control theory, and also an account of Logical Form "movement," perhaps interpretively, a la Aoun and Epstein (1988). It will also be of interest to study the implications of the more recent developments in the theory (Chomsky, 1986b) for the attribute grammar model, and to study the way in which the new developments may be captured within the attribute grammar framework.

In this thesis we considered English as the domain language. In spite of the importance of the research direction suggested above, another direction of great interest to pursue is the development of grammars for other natural languages within the attribute grammar framework. This move is certain to open new problems and issues in the description of language, for languages with different typological characteristics than English. For example, non-configurational languages will probably require careful consideration of the alternatives there may be to describe free word order. The transformational approach, assuming a scrambling rule, or an approach as Hale's (1983), in which a linking rule relates two levels of representation, one with rigid constituent order and another with essentially free constituent order, are not the only ones. An unordered context-free grammar for the base, similar to the immediate domination rules in GPSG, may be adequate. An unordered context-free grammar can always be reduced to a strongly equivalent context-free grammar, although with a larger production set. In this situation, the binary-branching rule approach of (Jensen, 1987) is of interest, since it requires at most duplication of the number of context-free productions needed. Attribute annotations could be added to compute "abstract syntax" trees as needed (DeRemer, 1976).

The new research suggested would permit comparison of the substantive elements in the grammars arrived at and identification of those places in the grammars where parameterization is

possible. Of special interest are the parameterization of the base, the identification and structuring of the attribute set involved, and the exploration of the extent to which the interpretive rules developed here, including the Chain and Binding rules, apply to other languages. Results of this kind can lead to an empirically-based attributed theory of grammar.

There are several technical directions in which the work done here should be extended. These have to do with the problem of mapping a given attribute grammar language definition into a parsing procedure and attribute evaluator, which can be used effectively for analysis. The method used here, which yields an LL(1) backtracking parser and an attribute evaluator, is perhaps the simplest such method.

We believe that an improvement of the parser can be obtained by the use of a parallel, non-backtracking recognition algorithm, such as Earley's (1968) algorithm, in place of the backtracking LL(1) algorithm used here. This change of algorithm would change the theoretical recognition time for the underlying context-free grammar from exponential, due to the use of backtracking, to polynomial. Since the grammar is attributed and the underlying grammar ambiguous, it is not clear what the overall effect on recognition time would be for the attribute grammar. Attribute evaluation must be performed in conjunction with construction of the tree or, in a simpler parsing model, after a tree is built. For the grammar of Theorem 1 in Chapter 3, which is infinitely ambiguous and simulates a Turing machine, some computations may not terminate. But it seems intuitively likely that the use of parallelism instead of backtracking in the parsing process will in general speed-up computation. The main reason is that reduplication of work is avoided.

An interesting problem left open in this thesis is the design of efficient attribute evaluators using unification and techniques of logic programming. We believe that the technique of Chapter 4 is a suitable basis, which can be improved considerably by making it applicable to a larger class of attribute grammars, without introducing non-determinism in the evaluators produced. Also, we believe the procedure should be automated once a suitable syntax is defined for attribute grammar notations. In this regard see (Watt and Madsen, 1983). Other work on attribute evaluation has concentrated on the identification, given a parse tree and possibly some other restrictions, of a sequence of tree traversals that may be used to evaluate all attributes in a tree (Bochmann, 1976; Kastens, Hutt, and Zimmermann, 1982; Watt and Madsen, 1983).

The assumption that the syntactic tree is already available, or that it can be computed independently of attribute evaluation is especially unrealistic in the context of natural language grammars. If the attribute grammar developed here for English is indicative of natural language grammars, we see that for natural languages the underlying context-free syntax may have a large (perhaps infinite) degree of ambiguity, and that attribute conditions play a role in reducing ambiguity, by rule-splitting (Watt, 1980). Hence, it is important to develop methods for attribute evaluation which may be used in attribute-directed parsing.

Several methods are available to transform a given attribute grammar definition into an equivalent one, but with certain other more desirable properties. This brings us back to the question of "grammars as specifications" which may be formally manipulated. An example of this is the "hoisting transformation" of (Waite and Goos, 1984), which is useful for adapting the grammar for a specified tree traversal. This transformation removes an inherited attribute of a symbol, along with synthesized attributes which depend on it, and replaces their semantic effect by a computation on the parent node. In (Correa, 1987b), an alternate version of the Chain rule is given, which is more suitable for bottom-up evaluation. In the version of the rule presented here, the attributes *A-Chain* and *AB-Chain* are inherited (propagated downwards) and the *Chain* attribute of a trace depends on the values of the two other attributes at the parent node. The Chain rule developed here is more suitable for top-down evaluation.

References

Abney, Steven. 1986. "Licensing and Parsing." *manuscript*, MIT.

Abney, Steven. 1987. "The English Noun Phrase in its Sentential Aspect." MIT Ph.D. Dissertation.

Abney, Steven, and Jennifer Cole. 1986. "A Government-binding Parser." *manuscript*, MIT.

Akmajian, Adrian, and Thomas Wasow. 1975. "The Constituent Structure of VP and AUX and the Position of the Verb BE," *Linguistic Analysis* Vol. 1, p. 205-246.

Aoun, Joseph. 1985. *A Grammar of Anaphora*, MIT Press. Cambridge, MA.

Aoun, Joseph, and S. Epstein. 1988. "A Computational Treatment of Quantifier Scope." To appear in CUNY Conference on Human Sentence Processing, March, 1988.

Barss, Andrew. 1983. "Chain Binding." *manuscript*, MIT.

Barton, Edward. 1984. "Towards a Principled-based Parser." *A.I. Memo No. 788*, MIT.

Barton, G. Edward, R. C. Berwick, and E. S. Ristad. 1987. *Computational Complexity and Natural Language.* MIT Press, Cambridge, MA.

Berwick, Robert and A. Weinberg. 1984. *The Grammatical Basis of Linguistic Performance.* MIT Press, Cambridge, MA.

Beale, Andrew D. 1985. "Grammatical Analysis by Computer of the Lancaster-Oslo/Bergen (LOB) Corpus of British English." *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois, p. 293-298.

Bresnan, Joan, ed. 1982. *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, MA.

Bresnan, Joan, and R. Kaplan. 1982. "Lexical Functional Grammar: A Formal System for Grammatical Representation." In *The Mental Representation of Grammatical Relations*, J. Bresnan, ed., MIT Press, Cambridge, MA.

Bochmann, Gregor. 1976. "Semantic Evaluation from Left to Right." *Communications of the ACM*, Vol.19, No. 2, p.55-62

Cherry, Colin. 1978. *On Human Communication: A Review, a Survey and a Criticism, Third edition.* MIT Press, Cambridge, MA.

Chomsky, Noam. 1955. *The Logical Structure of Linguistic Theory.* Plenum Press (1975), New York.

Chomsky, Noam. 1957. *Syntactic Structures.* The Hague, Mouton.

Chomsky, Noam. 1959. "On Certain Formal Properties of Grammar." *Information and Control, No. 2.*, p. 137-167.

Chomsky, Noam. 1964. *Current Issues in Linguistic Theory.* The Hague, Mouton.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax.* MIT Press, Cambridge, MA.

Chomsky, Noam. 1970. "Remarks on Nominalization." In R. Jacobs and P. S. Rosenbaum, eds., *Readings in English Transformational Grammar.* Waltham, MA.

Chomsky, Noam. 1977. *Essays on Form and Interpretation.* North Holland, New York.

Chomsky, Noam. 1980. "On Binding." *Linguistic Inquiry.* Vol. 11, p. 1-46.

Chomsky, Noam. 1981. *Lectures on Government and Binding.* Foris Publications, Dordrecht.

Chomsky, Noam. 1982. *Some Concepts and Consequences of the Theory of Government and Binding.* MIT Press, Cambridge, MA.

Chomsky, Noam. 1986a. *Knowledge of Language.* Praeger, New York.

Chomsky, Noam. 1986b. *Barriers.* MIT Press, Cambridge, MA.

Chomsky, Noam, and H. Lasnik. 1977. "Filters and Control." *Linguistic Inquiry.* Vol. 8, p. 425-504.

Church, Alonzo. 1941. *The Calculi of Lambda-Conversion.* Annals of Mathematical Studies 6. Princeton University Press, Princeton.

Clocksin and Mellish. 1981. *Programming in Prolog.* Springer Verlag, New York

Cooper, Robin. 1983. *Quantification and Syntactic Theory.* D. Reidel Publishing Company, Dordrecht.

Correa, Nelson. 1987a. "An Attribute Grammar Implementation of Government-binding Theory." *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics,* Stanford University, Stanford, California.

Correa, Nelson. 1987b. "Empty Categories, Chain Binding, and Parsing." Presented at parsing seminar, MIT Center for Cognitive Science, Dec. 2, 1987. To appear in *Working Papers of the Center for Cognitive Science,* MIT.

Correa, Nelson. 1988. "A Binding Rule for Government-binding Parsing." To appear in *Proceedings of the 12th International Conference on Computational Linguistics,* 22-27 August, 1988, Budapest, Hungary.

Deransart, Pierre, and J. Maluszynski. 1985. "Relating Logic Programs and Attribute Grammars." *The Journal of Logic Programming,* No. 2, p.119-155.

DeRemer, Franklin. 1976. "Review of Formalisms and Notation." In F. L. Bauer and J. Eickel, eds., *Compiler Construction: An Advanced Course.* Lecture Notes in Computer Science, Springer-Verlag, New York.

Dorr, Bonnie. 1987. "A Critique and Analysis of Current GB Parsing Systems." *manuscript,* MIT.

Earley, Jay. 1968. "An Efficient Context-free Parsing Algorithm." *Communications of the Association for Computing Machinery,* Vol. 13, No. 12, p. 94-102.

Ebbinghaus, H.D., J. Flum, and W. Thomas. 1984. *Mathematical Logic.* Springer-Verlag, New York.

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

Engelfriet, Joost and G. Filè. 1981. "The Formal Power of One-Visit Attribute Grammars." *Acta Informatica* No. 16, p. 275 - 302.

Farmer, Ann. 1984. *Modularity in Syntax: A Study of Japanese and English.* The MIT Press, Cambridge, MA

Fillmore, C. J. 1968. "The Case for Case." In E. Bach and R. Harms, eds., *Universals of Linguistic Theory.* Holt, Reinhart, and Winston, New York.

Fodor, Janet. 1978. "Parsing Strategies and Constraints on Transformations." *Linguistic Inquiry,* Vol. 9, p. 427-473.

Gazdar, Gerald, Geoffrey K. Pullum, and Ivan Sag. 1982. "Auxiliaries and Related Phenomena in a Restrictive Theory of Grammar," *Language*, Vol. 58, No. 3, p. 591-638.

Gazdar, Gerald, E. Klein, G.K. Pullum, and I. Sag. 1985. *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, MA.

Greibach, S. A. 1965. "A New Normal Form Theorem for Context-free, Phrase-structure Grammars." *Journal of the ACM*, 12, p., 42-52.

Gruber, Jeffrey. 1976. *Lexical Structures in Syntax and Semantics.* North-Holland, Amsterdam.

Hale, Kenneth. 1983. "Walpiri and the Grammar of Non-configurational languages." *Natural Language and Linguistic Theory* 1.

Hale, Kenneth and Jay Kayser. 1985. "Some Transitivity Alternations in English." *manuscript*, MIT.

Harrison, Michael. 1978. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, MA.

Heidorn, George. 1972. "Natural Language Inputs to a Simulation System." Naval Postgraduate School Technical Report No. NPS-55HD72101A.

Heidorn, George. 1975. "Augmented Phrase Structure Grammars." In B. L. Nash-Webber and R. C. Schank, eds., *Theoretical Issues in Natural Language Processing.* Association for Computational Linguistics.

Heidorn, George. 1976. "An Easily Computed Metric for Ranking Alternative Parses." Presented at the *Fourteenth Annual Meeting of the Association for Computational Linguistics*, San Francisco, CA.

Higginbotham, James and Robert May. 1981. "Questions, Quantifiers and Crossing." *The Linguistic Review 1*

Hoare, C. A. R. 1962. "Quicksort." *Computer Journal*, 5:1, p.10-15.

Hobbs, Jerry. 1978. "Resolving Pronoun References." *Lingua* No. 44, p. 311-338.

Hopcroft, Jeffrey and D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA.

Huang, C.-T. J. 1982. "Logical Relations in Chinese and the Theory of Grammar." MIT Ph.D. Dissertation.

Irons, Edgar. 1961. "Syntax-directed Compiler for ALGOL-60." *Communications of the ACM*, Vol. 4, No. 1, p. 51-55.

Jackendoff, Ray. 1972. *Semantic Interpretation in Generative Grammar.* MIT Press, Cambridge, MA.

Jackendoff, Ray. 1976. "Toward an Explanatory Semantic Representation." *Linguistic Inquiry 7*

Jackendoff, Ray. 1977. *X' Syntax: A Study of Phrase Structure.* MIT Press, Cambridge, MA.

Jackendoff, Ray. 1983. *Semantics and Cognition.* MIT Press, Cambridge, MA.

Jackendoff, Ray. 1987. "The Status of Thematic Relations in Linguistic Theory." *Linguistic Inquiry* Vol.18, No.3.

Jazayeri, Mehdi, W. Odgen, and W. C. Rounds. 1975. "The Intrinsically Exponential Complexity of the Circularity Problem for Attribute Grammars." *Communications of the ACM* Vol. 18, No.12., p.697-706.

Jazayeri, Mehdi. 1981. "A Simpler Construction Showing the Intrinsically Exponential Complexity of the Circularity Problem for Attribute Grammars." *Journal of the ACM* Vol. 28, No.4., p.715-720.

Jensen, Karen. 1986. "PEG 1986: A Broad-coverage Computational Syntax of English." IBM Thomas J. Watson Research Center, Yorktown Heights, NY.

Jensen, Karen. 1987. "Binary Rules and Non-binary Trees: breaking down the concept of phrase structure." In A. Manaster-Ramer, ed., *Mathematics of Language.* John Benjamin.

Jensen, Karen. 1988. "Issues in Parsing." In A. Blaser, ed., *Natural Language at the Computer: contributions to syntax and semantics for text-processing and man-machine communication.* IBM Deutschland GmbH, TR 88.02.002.

Johnson, Mark. 1987. "The Use of Knowledge of Language." *manuscript.* Brain and Cognitive Sciences, MIT.

Jones, Clifford. 1980. *Software Dervelopment: A Rigorous Approach.* North-Holland, Amsterdam.

Kashket, Michael. 1986. "Parsing a Free-word Order Language." *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics,* New York.

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

Kayne, Richard. 1984. *Connectedness and Binary Branching.* Foris Publications, Dordrecht.

Kimball, John. 1967. "Predicates Definable over Transformational Derivations by Intersection with Regular Languages." *Information and Control.* No. 11.

Kimball, John. 1972. "Seven Principles of Surface Structure Parsing in Natural Language." *Cognition*, Vol. 2, No. 1, p. 15-47.

Kiss, Katalin. 1987. "Is the VP Universal." In I. Keneseiled, ed., *Approaches to Hungarian*, Vol. 2. Szeged, Hungary.

Knuth, Donald. 1968. "Semantics of Context-free Languages." *Mathematical Systems Theory*, Vol. 2., No. 2, p. 127-145.

Knuth, Donald. 1974. "Estimating the Efficiency of Backtrack Programs." STAN-CS-74-442, Computer Science Department, Stanford University.

Kornai, Andras. 1983. "X-Bar Grammars." In J. Demetrovics, G. O. H. Katrona, and A. Salomaa, eds., *Algebra, Combinatorics, and Logic in Computer Science.* North-Holland, Amsterdam.

Kornfilt, Jaklin. 1984. "Case-marking, Agreement, and Empty Categories in Turkish." Harvard University, Ph.D. Dissertation.

Kornfilt, Jaklin, and N. Correa. In preparation. "Conceptual Structure and its Relation to the Structure of Lexical Entries."

Koster, C. H. 1971. "Affix Grammars." In *IFIP Working Conference on Algol 68 Implementation.* North-Holland, Amsterdam.

Koster, Jan. 1978. "Conditions, Empty Nodes, and Markedness." *Linguistic Inquiry*, No. 9, p.551-593.

Kuhns, Robert. 1986. "A PROLOG Implementation of Government-binding Theory." *Proceedings of the 1986 Annual Conference of the European Chapter of the Association for Computational Linguistics,*

Kuno, Susumu. 1987. *Functional Syntax: Anaphora, Discourse and Empathy.* The University of Chicago Press, Chicago, Illinos.

Langacker, R. 1969. "On Pronominalization and the Chain of Command." In D. A. Reibel and S. A. Schane, eds., *Modern Studies in English.* Prentice-Hall, Englewood Cliffs, NJ.

Langendoen, Terry. 1987. "On Phrasing of Coordinate Compound Structures." In A. Zwicky and D. Joseph, eds., Ohio State University Working papers in Linguistics, 35. 1987.

Lasnik, Howard. 1976. "Some Thoughts on Coreference." *Linguistic Analysis.* Vol. 2.

Levin, Beth. 1985. "Lexical Semantics in Review: an Introduction." In Lexicon Project Working Papers 1, ed. by Beth Levin. Center for Cognitive Science, MIT

Levin, Beth, and M. Rappaport. 1986. "The Formation of Adjectival Passives." *Linguistic Inquiry*. Vol. 17, No. 4.

Lewis. P. M., D. J. Rosenkrantz, and R. E. Stearns. 1974. "Attributed Translations." *Journal of Computer and System Sciences*, Vol. 9, p. 279-307.

Lindenmayer, A. 1968. "Mathematical Models for Cellular Interaction in development I-II." *Journal of Theoretical Biology 18*

Longman. 1981. *Longman Dictionary of Contemporary English.* The Pitman Press, Bath.

Madsen, O. Lehrmann. 1980. "On Defining Semantics by Means of Extended Attribute Grammars." In N. D. Jones, ed., *Semantics-directed Compiler Generation.* Lecture Notes in Computer Science, No. 94, p. 259-299, Springer-Verlag, New York.

Marcus, Mitchell. 1980. *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, MA.

Matthews, G. H. 1962. "Analysis by Synthesis of Sentences of Natural Languages." In *Proceedings of the 1961 International Congress on Machine Translation of Languages and Applied Language Analysis.* Teddington, England, National Physical Laboratory.

May, Robert. 1985. *Logical Form: Its Structure and Derivation.* MIT Press, Cambridge, MA.

Pereira, Fernando, and D. H. Warren. 1983. "Parsing as Deduction." *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, MIT, Massachussetts, p. 137-144.

Pesetsky, David. 1982. "Paths and Categories." MIT Ph.D. Dissertation.

Peters, Stanley, and R. Ritchie. 1973. "On the Generative Power of Transformational Grammars." *Information Sciences* 6.

Petrick, Stanley. 1965. "A Recognition Procedure for Transformational Grammars." MIT Ph.D. Dissertation.

Pollard, Carl, and I. Sag. 1987. *Information-Based Syntax and Semantics, Vol. 1: Fundamentals.* CSLI Lecture Notes 12, Stanford University, Stanford, CA.

Pullum, Geoffrey. 1985. "Assuming Some Version of the X-Bar Theory." *Manuscript.* Syntax Research Center, UC Santa Cruz.

Ravin, Yael. 1976. "A Decompositional Approach to Predicates Denoting Events." The City University of New York, Ph.D. Dissertation.

Reinhart, Tania. 1976. "The Syntactic Domain of Anaphora." MIT Ph.D. Dissertation.

Reps, Thomas. 1984. *Generating Language-based Environments.* ACM Doctoral Dissertation Awards, MIT Press, Cambridge, MA.

Ristad, Eric. 1986. "Computational Complexity of Current GPSG Recognition." *A.I. Memo No. 894*, MIT

Rizzi, Luigi. 1982. *Issues in Italian Syntax.* Foris Publications, Dordrecht.

Rizzi, Luigi. 1986. "Null Objects in Italian and the Theory of pro." *Linguistic Inquiry.* Vol. 17, No. 3.

Robinson, Alan. 1965. "A Machine-oriented Logic Based on the Resolution Principle." *Journal of the Association for Computing Machinery.* No. 12, p. 23-41

Robinson, Alan. 1979. *Logic: Form and Function. The Mechanization of Deductive Reasoning.* North-Holland, Amsterdam.

Ross, John. 1967a. "Constraints on Variables in Syntax." MIT Ph.D. Dissertation.

Ross, John. 1967b. "Auxiliaries and Main Verbs," *Studies in Philosophical Linguistics.* Great Expectations Press, Evanston, Illinois.

Sells, Peter. 1985. *Lectures on Contemporary Linguistic Theories.* Chicago University Press, Chicago, Illinois.

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

Shieber, Stuart. 1986. *An Introduction to Unification-based Approaches to Grammar.* CSLI Lecture Notes No. 4, Stanford University, Stanford.

Sonnenschein, Michael. 1985. "Global Storage Cells for Attributes in an Attribute Grammar." *Acta Informatica.* No. 22, p. 397 - 420.

Steele, Susan. 1981. *An Encyclopedia of AUX: A Study in Cross-Linguistic Equivalence.* Linguistic Inquiry Monograph Five. The MIT Press, Cambridge, MA

Sterling, Leon, and E. Shapiro. 1986. *The Art of Prolog: Advanced programming techniques.* MIT Press, Cambridge, MA.

Stowell, Timothy. 1981. *Origins of Phrase Structure.* MIT Ph.D. dissertation.

Stoy, Joseph. 1977. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* MIT Press, Cambridge, MA.

Talmy, Leonard. 1985. "Lexicalization Patterns: Semantic Structure in Lexical Forms." *In Language Typology and Syntactic Description 3.* Cambridge University Press, Cambridge.

Tamaki, Hisao, and T. Sato. 1984. "Fold/Unfold Transformation of Logic Programs." In *Proceedings of the Second International Logic Programming Conference.* Uppsala University, Uppsala, Sweden, p. 127-138.

Travis, Lisa. 1984. "Parameters and Effects of Word Order Variation." MIT Ph.D. dissertation.

van Riemsdijk, Henk and Edwin Williams. 1986. *An Introduction to the Theory of Grammar.* The MIT Press, Cambridge, Massachussets.

van Wijngaarden, A. 1974. "The Generative Power of Two-level Grammars." In *Automata, Languages, and Programming.* Lecture Notes in Computer Science, Springer-Verlag, New York.

Waite, William M. and Gerhard Goos. 1984. *Compiler Construction.* Springer-Verlag, New York.

Wasow, T. 1972. "Anaphoric Relations in English." MIT Ph.D. dissertation.

Watt, David. 1980. "Rule Splitting and Attribute-directed Parsing." In N. D. Jones, ed., *Semantics-directed Compiler Generation.* Lecture Notes in Computer Science, No. 94, p. 363-392, Springer-Verlag, New York.

Watt, David, and O. L. Madsen. 1983. "Extended Attribute Grammars." *The Computer Journal.* Vol. 26, No. 2, p.142-153.

Webster's Seventh Collegiate Dictionary. 1967. C. & C. Merriam Company. Springfield. MA.

Wehrli, Eric. 1984. "A Government-binding Parser for French." Institut Pour les Etudes Semantiques et Cognitives, Universite de Geneve. Working Paper No. 48.

Williams, Edwin. 1977. "Discourse and Logical Form." *Linguistic Inquiry 8*

Williams, Edwin. 1981. "Argument Structure and Morphology." *The Linguistic Review, No. 1*

Williams, Edwin. 1984. "Grammatical Relations." *Linguistic Inquiry*, Vol. 15.

Yeh, Dashing. 1983. "On Incremental Evaluation of Ordered Attribute Grammars." *BIT*, 23, p. 308-320.

Appendix: Sample Parser Runs

1. Simple declarative sentences

```
------------------------------------------------------------------
gbp.
 GOAL :  <- gbp .
Type in an input CLAUSE: the man sees Mary.
PARSING ...
Parse Time: 13 mSec.
Parse:
CP   C'   C   -[]
          IP   NP5  ArtP Art' Art  the[]
                    N'   N    man[]
               I'   I   -[]
                    VP   V'   V    sees[]
                              NP16 N'   N    Mary[]

 Find another (y/n) ?  y
PARSING more ...       Parse Time: 175 mSec.     No (more) parses found.

------------------------------------------------------------------
Type in an input CLAUSE: the man thinks he sees Mary.
PARSING ...
Parse Time: 27 mSec.
Parse:
CP   C'   C   -[]
          IP   NP5  ArtP Art' Art  the[]
                    N'   N    man[]
               I'   I   -[]
                    VP   V'   V    thinks[]
                              CP   C'   C   -[]
                                        IP   NP20 N'   N    he[]
                                             I'   I   -[]
                                                  VP   V'   V    sees[]
                                                            NP28 N'   N    Mary[]

 Find another (y/n) ?  y
PARSING more ...       Parse Time: 194 mSec.     No (more) parses found.

------------------------------------------------------------------
```

2. NP-movement sentences

```
------------------------------------------------------------
Type in an input CLAUSE: the man was seen.
PARSING ...
Parse Time: 148 mSec.
Parse:
CP   C'   C   -[]
          IP   NP5  ArtP Art' Art  the[]
                    N'   N    man[]
               I'   I    MP
                         HP
                         TP
                         B1P
                         B2P  B2   was[]
                    VP   V'   V    seen[]
                              NP5-[]

 Find another (y/n) ?  y
PARSING more ...       Parse Time: 138 mSec.     No (more) parses found.

------------------------------------------------------------
Type in an input CLAUSE: the man is believed to be seen.
PARSING ...
Parse Time: 201 mSec.
Parse:
CP   C'   C   -[]
          IP   NP5  ArtP Art' Art  the[]
                    N'   N    man[]
               I'   I    MP
                         HP
                         TP
                         B1P
                         B2P  B2   is[]
                    VP   V'   V    believed[]
                              CP   C'   C   -[]
                                        IP   NP5-[]
                                             I'   I    MP
                                                       HP
                                                       TP   TO   to[]
                                                       B1P
                                                       B2P  B2   be[]
                                                  VP   V'   V    seen[]
                                                            NP5-[]

 Find another (y/n) ?  y
PARSING more ...       Parse Time: 160 mSec.     No (more) parses found.
------------------------------------------------------------
```

3. Relative clauses

```
-----------------------------------------------------------------
Type in an input CLAUSE: the man who sees Mary came.
PARSING ...
Parse Time: 28 mSec.
Parse:
CP   C'   C   -[]
          IP   NP5  ArtP Art' Art  the[]
                    N'   N    man[]
                         CP   NPi2 N'   N    who[]
                              C'   C   -[]
                                   IP   NP12-[]
                                        I'   I   -[]
                                             VP   V'   V    sees[]
                                                       NP24 N'   N    Mary[]
               I'   I   -[]
                    VP   V'   V    came[]

 Find another (y/n) ? y
PARSING more ...      Parse Time: 279 mSec. No (more) parses found.

-----------------------------------------------------------------
Type in an input CLAUSE: the man who Mary sees came.
PARSING ...
Parse Time: 27 mSec.
Parse:
CP   C'   C   -[]
          IP   NP5  ArtP Art' Art  the[]
                    N'   N    man[]
                         CP   NP12 N'   N    who[]
                              C'   C   -[]
                                   IP   NP18 N'   N    Mary[]
                                        I'   I   -[]
                                             VP   V'   V    sees[]
                                                       NP12-[]
               I'   I   -[]
                    VP   V'   V    came[]

 Find another (y/n) ?  y
PARSING more ...      Parse Time: 300 mSec.     No (more) parses found.
-----------------------------------------------------------------
```

4. Wh-movement sentences

```
------------------------------------------------------------------
Type in an input CLAUSE: who does the man think sees Mary.
PARSING ...
Parse Time: 59 mSec.
Parse:
CP   NP2  N'   N    who[]
     C'   C    I    DO   does[]
          IP   NP10 ArtP Art' Art  the[]
                    N'   N    man[]
               I'   I   -[]
                    VP   V'   V    think[]
                              CP   NP2-[]
                                   C'   C   -[]
                                        IP   NP2-[]
                                             I'   I   -[]
                                                  VP   V'   V    sees[]
                                                            NP32 N'   N    Mary[]

 Find another (y/n) ?  y
PARSING more ...       Parse Time: 165 mSec.     No (more) parses found.

------------------------------------------------------------------
Type in an input CLAUSE: who does the man think Mary sees.
PARSING ...
Parse Time: 61 mSec.
Parse:
CP   NP2  N'   N    who[]
     C'   C    I    DO   does[]
          IP   NP10 ArtP Art' Art  the[]
                    N'   N    man[]
               I'   I   -[]
                    VP   V'   V    think[]
                              CP   NP2-[]
                                   C'   C   -[]
                                        IP   NP26 N'   N    Mary[]
                                             I'   I   -[]
                                                  VP   V'   V    sees[]
                                                            NP2-[]

 Find another (y/n) ?  y
PARSING more ...       Parse Time: 171 mSec.     No (more) parses found.

------------------------------------------------------------------
```

5. NP- and Wh-movement

```
-----------------------------------------------------------------
Type in an input CLAUSE: who does the man seem to love.
PARSING ...
Parse Time: 140 mSec.
Parse:
CP NP2  N'  N    who[agr(3,*1,sing)]
   C'   C   I    DO   does[tense(+)]
        IP  NP10 ArtP Art' Art  the[agr(3,masc,sing)]
                 N'  N    man[agr(3,masc,sing)]
            I'   I  -[tenseS(-)]
                 VP  V'  V    seem[agr(0,0,0),tense(-)]
                         CP   NP2-[gov(24,proper),case(nom)]
                              C'   C   -[tense(-)]
                                   IP   NP10-[gov(20,proper),case(nil)]
                                        I'  I   TO  to[tense(-)]
                                            VP  V'  V    love[agr(0,0,0),tense(-)]
                                                    NP2-[gov(37,proper),case(acc)]

  Find another (y/n) ?  y
PARSING more ...       Parse Time: 115 mSec.     No (more) parses found.
-----------------------------------------------------------------
```

Biographical Data

Name:	Nelson Correa
Date and Place of Birth:	May 10, 1958 El Centro, Santander, Colombia
Education:	
Elementary school	Colegio Luis López de Mesa El Centro, Santander, Colombia
Secondary school	Instituto de La Salle Bogotá, Colombia
University	Universidad Javeriana Bogotá, Colombia B.S., 1979
	Syracuse University Syracuse, New York Research and Teaching Assistant M.S., 1981 M.A., 1985