Description Theory, Licensing Theory, and Principle-Based Grammars and Parsers

Thomas Longacre Cornell

1992

To my parents,
Louis and Sara,
who did more than anyone else
to make this possible

Contents

List of Figures

ACKNOWLEDGMENTS

A project of this scope necessarily owes much to the efforts of many people beyond the author. First and foremost, thanks go to my committee: to my co-chairs Ed Stabler and Vicki Fromkin for overall guidance, and essential insights into the fields in which they were my teachers, to Dominique Sportiche, whose systematic grasp of syntactic theory was (and is) an inspiration to me, to Drew Moshier, who first taught me the logic of feature structures, and to Randy Gallistel and Keith Holyoak, whose interest and curiosity (and patience with my fumbling explanations!) concerning matters of fairly hard-core linguistics has greatly encouraged my faith in the essential unity of the cognitive science community. [1] My only regret is that, with such a collection of intellects, the only times I could get them all together in one room, I had to do so much of the talking—I would much rather have listened.

Besides the members of my committee, special thanks are due to Susie Curtiss and Ed Keenan, whose influence on the course of this work, at various crucial junctures, has been enormous. Between them they made so much possible for me that would have otherwise been impossible.

Thanks go also to my many teachers at UCLA.Eric Wehrli, Robin Clark and Ed Stabler taught me computational linguistics, and to them I owe a special debt of gratitude. In addition, Nina Hyams, Tim Stowell, Hilda Koopman and Steve Anderson all played very large roles in shaping the way I think about language and linguistics.

I have also learned a lot from discussions with my fellow students of computational linguistics. Bill Dolan, Jim Tyhurst, Andi Wu and Susan Hirsch deserve special mention. Also, the psycholinguistics working group at UCLA gave me a forum for half-baked ideas, and tremendous moral support—I owe them a special debt of gratitude, in particular to Karen Emmorey, Jill Langley, Zohar Eviatar, Jack Dwyer, and, most especially, Gail Mauner. My long collaboration with Gail and Vicki Fromkin was, I think, the intellectual high water mark of my graduate career.

For much of the last year, I have called the University of Arizona home, and I owe the people here a particular debt of gratitude for putting up with the time it took me to finish this project. In the psycholinguistics lab, my new home, special thanks are due to Merrill Garrett and Ken Forster, who've heard me out on many of the topics covered here, and much, much else, patiently and helpfully. Also, thanks to Janet Nicol, Andy Barss and Paul Bloom. In the linguistics department at Arizona, Terry Langendoen and Dick Oehrle have been very supportive, encouraging and helpful. In addition,

[1] Unfortunately, Keith could not be present at my defense, and I feel certain we all missed his voice in the discussion there.

special thanks are due to Jo Calder, my comrade in computational linguistic arms.

I have at various times in the course of this project benefitted from discussions with a number of researchers in the fields covered here. Special thanks, in this department, go to Mark Johnson, who listened closely to my ramblings about this topic and made a number of very useful suggestions. I also benefitted greatly from a few (too few!) days in Nijmegen in long and deep discussion with Henk Haarman, for which I remain very grateful. In addition, Stott Parker, David Lebeaux, Jennifer Cole and Yosi Grodzinsky deserve my thanks for good advice and encouragement.

Theirs the credit, mine the fault.

I would also like to thank some of the many people who made my life so much more pleasant during these years of toil. Besides many friends already mentioned, Chris Golston, Cheryl Chan, Harold and Muna Crook, Yasushi Zenno, Sue Inouye, Sigga Sigurjónsdóttir, Bonnie Chiu, Karen Wallace, Mike Dukes, Bia Amos, Lukowa Kidima, Hyunoo Lee, and Jack Fromkin deserve special mention. Deluxe extra-special thanks are due to Emily Sityar, who put up with me and pulled me through the darkest times. I hope and pray I will be able to return the favor.

All'alta fantasia qui mancò possa;
ma già volgeva il mio disio e 'l velle,
sì come rota ch' igualmente è mossa,
l'amor che move il sole e l'altre stelle.

"Here power failed the high phantasy; but now my desire and will, like a wheel that spins with even motion, were revolved by the Love that moves the sun and the other stars." [trans., John D. Sinclair]

Abstract

In this dissertation I will lay the foundations for an approach to principle-based parsing which has some interesting theoretical consequences, among them a much clearer picture of the place of GB Theory within the wider universe of formal frameworks. I will approach parsing with systems of grammatical principles via the simpler problem of parsing with systems of licensing conditions. I will develop a formalism for expressing Licensing Grammars, that is, grammars made up entirely of licensing conditions, and explore the interpretation of such grammars as systems of constraints (their traditional interpretation) and as, in effect, systems of productions, treated like rewrite rules. As part of this exploration I will sketch a (partly implemented) computational model of a grammar interpreter for LGs.

An examination of the logic underlying licensing, and the notion of well formedness with respect to a system of licensing conditions, reveals that it is possible to treat licensing as a species of node expansion. Conversely, we can look at context-free grammars, for example, as special cases of licensing grammars. In order to assure that this approach extends to GB Theory, it is necessary to employ rather different representations of syntactic structure, namely logic-based representations which are collections of (atomic) constraints on nodes and features. These *logical descriptions* allow us to extract structure directly from licensing conditions, even though such conditions are often not very informative about the details of local immediate constituent relations. The hard problems of principle-based parsing are localized in the problem of combining logical descriptions, specifically in the *satisfiability problem* for logical descriptions. This problem remains hard, but this framework makes a little clearer how it can be approached.

Altogether, this computational framework, from specific accounts of the way in which licensing conditions effect chain formation, to general concepts governing the way in which logical descriptions relate to syntactic trees, gives us a way of dealing with language breakdown and its effects on linguistic computations, as I will show here by applying the concepts of Licensing Theory and Description Theory to problems of modeling comprehension and acceptability judgments of certain agrammatic aphasics.

Chapter 1

Introduction

The shift in the Chomskyan paradigm of syntactic theory to a Principles and Parameters framework [Chomsky 1981, Chomsky 1986a, and refs.] offers challenging and interesting problems to the study of computational linguistics. Many of these problems are foundational in nature and derive from profound shifts in the conception of grammar, which reached a watershed in Government-Binding (GB) Theory [Chomsky 1981]. In this work Chomsky proposed that Universal Grammar (UG) could be entirely characterized as a highly interactive system of *substantive universals*, the "principles" to which the label "Principles and Parameters" refers. These are normally expressed as constraints on what counts as a well-formed syntactic structure, that is, as *well-formedness conditions*.

A striking characteristic of the various theories which have evolved in the decade since [Chomsky 1981] is the prominence of *licensing conditions* among these well-formedness conditions. This thesis will offer an investigation of systems of licensing conditions from a computational perspective. My approach here will be to use GB Theory to help define what a licensing condition is, and then, definition in hand, to examine systems of licensing conditions in general, and GB as an instance of this more general framework. Conducting this investigation in such a way yields a better understanding of the place of principle-based grammars in the universe of formal grammars, in particular their relation to more classical phrase-structure grammars. As a matter of theoretical interest, such a demonstration is important in helping us to express more clearly what aspects of GB Theory are truly innovative. As a matter of practical interest, because phrase-structure grammars (in particular, context-free grammars) are well understood computationally, it is to be expected that establishing a connection from phrase-structure grammars to principle-based grammars will allow us to bring well understood computational techniques to bear on applications of principle-based grammars.

A licensing condition will be taken here to be a well-formedness condition on syntactic structures roughly of the form "an element of type F (the *focus* of the condition) is licensed to appear in a syntactic structure if it appears in a structural configuration E (the *environment* of the condition)." So for instance the Theta Criterion has as its consequence that an NP is licensed in d-structure if it appears as a sister to a thematic role assigning predicate, and the predicate assigns a role to the NP. That is, F would pick out nodes of category NP, and E would describe a strict sisterhood configuration with a certain identification of features of the argument NP with features of the predicate. Similarly, we could take F to refer to traces of moved elements, and E to refer to an antecedent-government configuration, giving us the Condition on Chain Links of [Sportiche 1990], for example. Note that licensing conditions typically license *nodes* in these examples. Note also that several licensing conditions could apply to a single node. For instance the trace of a nominal WH-phrase might have to receive a θ-role, and abstract Case as well, and in addition be antecedent governed, in order to be fully licensed.

Let us define what it means for a syntactic structure to satisfy a system of licensing conditions as follows. A structure satisfies a collection of licensing conditions if (a) all its nodes are licensed by all applicable conditions (the condition of Full Licensing) and in addition (b) all its nodes are licensed by at least one condition (the condition of Proper Licensing). If the grammar contains only licensing conditions then this definition of satisfaction is synonymous with the definition of well-formedness with respect to a grammar. Though this characterization of well-formedness may at first seem unfamiliar to GB syntacticians, condition (a) just expresses what it means to satisfy a system of implications, and condition (b) is just a restatement of a condition which appears in [Chomsky 1986a, p. 93], that all elements must be licensed.[1] The condition of Proper Licensing is required since Full Licensing is vacuously satisfied by an element to which no condition applies, that is, which is not described by any F. The Full Licensing condition is necessary since, as in the example of the nominal WH-trace, several conditions may apply to the same element. If any of these is violated, in the sense that F describes an element, but it is not in the environment specified in E, then the structure in which that element occurs is ill-formed to some degree, no matter how many other conditions that element succeeds in satisfying.[2]

Licensing in GB Theory is always *configurational*—every licensing condition contains some expression E, as in the above definition, which de-

[1] See also [Speas 1991], who also cites [Kim 1988].

[2] Cf. also [Macías 1990], who proposes similar definitions to handle licensing in a logical approach to GB parsing.

scribes some structural assertion about the tree. By this we mean, roughly, that E is an expression built somehow from the primitive building blocks of dominance and precedence, the relations which define syntactic structures. Understanding this is the key to relating systems of licensing conditions to classical phrase structure grammars such as context-free grammars (CFGs). A rewrite rule such as $S \longrightarrow NP\ VP$ is clearly configurational: it expresses, in a sense, the structure of one type of building block from which syntactic trees may be constructed, or into which a syntactic tree may be decomposed. In order to understand the relation between GB Theory and CFGs, then, we must find a way to relate the specification of structure in traditional rewrite rules to the way that structure is specified in licensing conditions. It will turn out that the framework for expressing licensing grammars which I will develop in what follows is sufficiently powerful to express CFGs as well, and yet sufficiently restrictive to throw into relief the chief features in which GB differs from CFGs.

Aside from this theoretical (or meta-theoretical) goal of understanding the connections of GB to earlier work and appreciating the innovations which chiefly characterize it, there is another, more practical goal. Since licensing conditions are configurational, that is, since they contain information about the structures in which they apply, we should expect to be able to extract from a licensing condition the structure that it specifies, once we know that it must apply to a particular element. This suggests an approach to principle-based parsing which stresses efficiently extracting this information and efficiently combining it with information already assumed about the structure of the input sentence.

The problem which faces all work in principle-based parsing is this. If we think of GB as a system of constraints, then the parsing problem has as its most natural formulation "Find the structure which satisfies all the constraints." Apparently this leads us to some kind of a generate-and-test algorithm, where we construct a phrase-marker, and then test to see if it is well-formed, continuing this procedure until we have found all the structures which pass this test. A problem with this approach is that, since the grammar is now completely made up of constraints, it belongs in the test stage, so that the generation stage is essentially random, and must navigate an infinite space of possible syntactic structures without a guide. The solution to such problems, as is widely known, is to interleave the generate and test steps as tightly as possible, so that the knowledge contained in the test procedure is brought to bear on generation as early as possible, and as little generation as possible is done before testing occurs.

The algorithms used to parse context-free grammars in effect collapse the generate and test steps so that grammatical knowledge is brought to bear

on every step of structure generation. I will show in this thesis how this kind of collapsing of generation and testing can be effected for at least some Licensing Grammars, that is, systems of licensing conditions. If we pursue this goal, of extracting structure directly from licensing conditions, then we are lead to a family of parsing algorithms which offer considerable hope for the development of efficient principle-based parsers.

The basic strategy is this. In what follows I will develop a means for expressing structural conditions, or structural information, with sets of atomic assertions, statements to the effect that "node α dominates node β," for example. Note that we can take such a statement as being a constraint: "whatever we take α to be, it must dominate β, or else!" On the other hand we can also take it as a piece of a tree, a path from whatever node α stands for to whatever node β stands for. In fact we can specify any particular tree in this way, as a set of such statements. If we can express the foci and environments of licensing conditions in the same way, as sets of atomic statements, then extracting structure from them is correspondingly simple. I will argue that much of current GB Theory can in fact be expressed this way.

The result, it should be kept in mind, is not in itself the grand solution to the problems of parsing with highly interactive systems of licensing conditions and hence, *a fortiori*, not the solution to the "GB Parsing Problem". The problem of correctly mapping the α's and β's that appear in licensing conditions to particular nodes in the tree being built is still a hard problem. However, the rigorously formal, logical approach which I will pursue here at very least allows us to state the problem in clear enough terms that the next steps to take are correspondingly clear, and solutions to many difficulties appear, if only in outline, and at a distance. Given a problem of the magnitude and complexity of Principle-Based Parsing, I think this is all a single dissertation can hope to achieve.

1.1 What does it mean to be *Principle Based*?

The shift from the standpoint of original Transformational Grammar to that of the Principles and Parameters framework is generally referred to as a shift from systems of "rules" to systems of "principles". Here I will try to give some idea of what I think this change entails. Grammatical frameworks (in particular the various versions of Transformational Grammar) which are often termed "rule-based" share the property that they begin with the specification of a formalism within which the statements which make up the body of that theory are to be rendered. The outlines of these formalisms (from [Chomsky 1955/75] to [Pollard 1987] and beyond) are almost invariably drawn from work in the theory of computation or, more recently, from work specifically in computational linguistics. The Principles and Parameters framework in no way abandons this stance, that language is a computational system (cf. [Chomsky 1986a, p. 43]). We may assume, then, that the change to principle-based grammars is *computational* in nature, and that a computational approach (rather than, say, a strictly descriptive/empirical one) is appropriate for investigations at this level.

The formal distinction between "rules" and "principles" is not always clear, and has proved very tricky to pin down satisfactorily. For example, Berwick [Berwick 1987, Berwick 1991] attempts to define principles as non language particular, non construction specific statements and rules as language-particular and construction specific. It is fairly clear, however, that once researchers began to propose that UG included as substantive universals conditions governing the application of transformational rules (cf. [Chomsky 1964]), the cross-constructional generality of these rules was correspondingly increased, though they clearly remained statements we would want to call "rules". Ultimately it is really just a question of doing good science that statements at the level of principle should be 'non-phenomenon-specific', and so this should never be a distinguishing characteristic of one framework.[3] On the other hand there is a clear methodological difference, namely that rules, at least transformational rules, were subject to *formal universals*, formal constraints, given *a priori*, governing what could be a possible transformation, and this has remained a characteristic of all recently proposed formal syntactic frameworks, except for GB. The *substantive universals* which, since the early 1960s began to be hypothesized of UG, were never, to the best of my knowledge, placed under any such constraints.[4] As a result, with the working hypothesis of [Chomsky 1981], that core grammar could be *entirely* expressed in terms of substantive universals, that is, in effect, without statements constrained by a formalism given *a priori*, work in Chomskyan theoretical linguistics moved to methods of theory construction which have set it apart, and contributed to much confusion about its place within the larger community of formal linguistics, in the minds both of practitioners and non-practitioners.

Nowhere have the conceptual, foundational difficulties engendered by this departure caused more problems than in the study of computational linguistics. As a symptom of the problem, it is interesting to note that only within the Principles and Parameters framework is there, or could there be, any significant debate over what counts as a parser within that framework. There is, in fact, a not inconsiderable literature on what counts as a principle-based parser [Berwick 1987, Berwick 1991] or a GB parser [Wehrli 1987]. No comparable questions arise in the face of a claim that X is a parser for GPSG, for example. The reason for this, I believe, is the absence of a GB formalism, given *a priori* and specifying what can be a GB principle or a GB grammar and what cannot. There seems to be little more constraint on the form of a principle than that reasonable people should not differ about its empirical consequences, and that if some proposal makes a particular prediction in Cambridge, Massachusetts, the same consequence must be derivable in Cambridge, England.

In practice this condition of rigorous interpretation means that most proposals, if not all, are susceptible of a straightforward logical interpretation. In fact, first order logic seems to be powerful enough to axiomatize GB theory [Stabler 1992, Johnson 1989], and this formalization is argued to be perfectly straightforward and to represent a transparent implementation of linguists' proposals [Stabler 1992]. Still it is significant that this formalization is *a posteriori*. This is unlike parsing work in most theories of grammar; it is unusual to have to formalize a theory before beginning to discuss the problems of parsing with it. It also makes the difficulty of building efficient parsing applications based on the theory that much greater, since first order logic is a formalism with intractable computational characteristics. In order to rein in these difficulties it is generally important to find features of a specific problem which set it apart from the general case and which one can capitalize on to parse more efficiently. But in this case it is so difficult to say what a GB theory is, other than by listing examples, that it is correspondingly hard to say what a GB parser is, and therefore what features it might possess which would make it run more efficiently than, say, a general purpose first-order theorem prover.[5]

It is important to keep in mind that this is not just a problem of building computer programs that run fast. The problem of formalizing a theory is an analytical problem involving foundational assumptions about the nature of language, grammar, grammaticality and other syntactic properties. The formalisms underlying most variants of generative grammar (which I

[3] Note that this requirement on statements "at the level of principle" does not, of course, preclude formulæ which are not principles. Presumably the rules of a GPSG grammar (for example) should not be chosen for comparison to GB principles any more than we should consider lexical entries to be principles. Indeed, [Berwick 1991] notes this, that a number of theoretical frameworks besides GB can be characterized as principle-based.

[4] This is also true of principles as they appear in other frameworks, such as GPSG or HPSG, which mix formalized rules with principles constraining the use of the formalism.

[5] The problem here is most acute if we are considering the development of *general* GB parsers, that is, parsers whose design and implementation is independent of any particular variant of the theory.

interpret loosely to encompass all work in formal linguistics descended on whichever side of the bed from Chomsky's early work [Chomsky 1955/75, Chomsky 1957]) are meant to address these issues in formal, logically rigorous terms, under the general working hypothesis that if grammars have structure this is unlikely to be accidental, and accordingly this structure should be taken to reveal something of the structure and organization of language.[6] Therefore it should in principle be an interesting *linguistic* study, and certainly important within the more general concerns of cognitive science, to examine the logical foundations of the principles and parameters framework in the light of computational formalization. Note however that there is no guarantee that there *is* such an object as the logical foundation of the principles and parameters framework—it may be, for instance, that the GB Community is more sharply defined than is GB Theory. In this dissertation I will argue that this is not the case, and that there are in fact interesting computational features underlying current principle-based grammars, and that these features can be of use in developing principle-based parsers. In particular the fixed logic underlying all licensing conditions can serve as a lever with which we can gain a computational advantage.

1.2 Licensing Grammars and Logical Descriptions

In this dissertation I will approach the more general topic of principle-based grammars and parsers by exploring properties of licensing conditions, grammars of licensing conditions (*licensing grammars*), and parsers which implement licensing grammars (*licensing parsers*). I will develop a formalism for expressing licensing grammars and suggest how it can be used to capture significant chunks of GB theory, as well as how a parser can most naturally apply the knowledge represented in those conditions.

The Licensing Grammar formalism I will develop is based on a first order logical language for expressing those properties of syntactic structures which play a significant role in the definition of linguistic principles. In addition I will provide this system with a model-theoretic "semantics" which interprets expressions of the logical description language in terms of tree structures, together with a sound and complete deductive calculus so that it can be made clear what the consequences are of any particular set of licensing conditions.

[6] This is not to say that the structure of the grammar has been of no interest to GB theorists. In fact, the modular organization of principles has received a good deal of attention in the literature.

Based on this system I will be able to develop a logical characterization of key concepts like grammaticality and syntactic well-formedness, from which, in turn, I will develop a model of a parsing system, a system which uses the concept of well-formedness with respect to a licensing grammar to construct a syntactic analysis of input sentences. The resulting model, which consists of a *grammar interpreter* for Licensing Grammars, captures the most natural way to implement the ideas developed here. Little attention is paid here to efficiency, and it is worth noting that, as is often the case, the most natural implementation is not strikingly efficient. However, in many respects it is clear how immediate progress can be made in this particular application of the framework developed here.

Besides giving us a language in which to write licensing grammars, the logical description language I will develop gives us a way to characterize particular syntactic representations, indirectly. That is, certain sets of ground, atomic formulas in the description language have unique *minimal canonical tree-models*, a concept I will develop in Chapter 2. As such they can stand for tree structures, though they are technically speaking only sets of constraints on such structures. This offers a way of parsing with a licensing grammar *indirectly*, by deriving one of these special collections of formulas rather than by constructing the tree which is its least canonical model. This has important benefits, for the following reasons.

One feature which sets apart the licensing conditions in GB from phrase structure rules in other grammars is that any given grammatical constraint can refer to substructures of the syntactic tree which are not themselves trees. The principle culprit here is the use of remote, transitive structural relations (i.e., dominance and precedence) rather than the immediate, intransitive structural relations which characterize phrase structure grammars. As a result, if we try to recover structure directly from the application of a licensing condition, we cannot represent the structure recovered in this way as a tree. Furthermore, such structures do not combine in the usual way, that is, by attachment of one (completely specified) subtree into another. (Here we think of rewriting S as NP plus VP as attaching an NP subtree under S, then attaching a VP subtree. This approach to tree-building from a grammar is made more explicit in Tree Adjoining Grammar [Joshi 1985], but is a fruitful way of looking at how a parser can build structure by reference to a rewrite grammar, as well.) Rather, the kinds of grammatical substructures specified in GB-style licensing conditions overlap each other, and only gradually fill in the immediate constituent structure assigned to a sentence. It is, of course, quite easy to represent structures based on remote structural relations using logic—we merely employ predicates which are interpreted as remote structural relations and axiomatize them so that they have the required properties of transitivity, etc. Combining such structures amounts to performing a set union, subject to certain logical consistency conditions. Thus, parsing with *logical descriptions* seems to be a very natural way of parsing with the kind of licensing conditions which appear in GB theory.

1.3 Partial Descriptions and Language Breakdown

While some descriptions are *complete* and can be used to pick out unique tree structures, the rest are *partial*. However, we can still extend a partial description to one or more complete ones, so in turn we can treat partial descriptions as standing in a sense for the set of such completions. The addition of structural underspecification increases our representational resources, placing them somewhere between trees and (arbitrary) sets of trees. Given these new expressive resources, and a correspondingly weaker theory of representation we must address the obvious next question: are these new resources useful?

I will argue that they are useful in modeling certain kinds of language breakdown. I will argue that certain kinds of brain damage lead to a loss of *resolution* in the syntactic processor, that the deficit leads to a loss of *information* at the syntactic level of representation. This amounts to the hypothesis that the processor is only arriving at a partial syntactic representation, and requires that we know in detail what we mean by such representations, so that we can say how processing can proceed using them.

I will address a specific cluster of symptoms which arises in some, perhaps most, agrammatic Broca's aphasics. The problem I will address involves difficulties which these subjects have in comprehending sentences which involve certain syntactic constructions. Though the generalizations involved in describing this behavior are clearly syntactic in nature, these subjects show a surprising ability to judge these same constructions in a grammaticality judgment task. I will attempt to elaborate a particular hypothesis of this deficit, and pursue its consequences within the Licensing Grammar formalism. Solving this problem of abnormal linguistic performance leads, I believe, to a much richer conception of Chain Theory than the study of normals has so far led us to. In particular, dividing the properties of chains up between their structural properties (in particular conditions on antecedent government) and their agreement, or feature-sharing properties puts us in a position to show how agrammatics' damaged parsers could build structures detailed enough to support accurate grammaticality judgments, but lacking

pieces essential for comprehension. I am certainly not the first researcher to propose that agrammatics construct somehow underspecified syntactic representations.[7] However, without a clear idea of what a partial representation is, it is hard to say how processing should proceed, in particular, how such structures could ever support accurate acceptability judgments, and so it is also hard to see clearly what the consequences of such a hypothesis are. I will demonstrate that once we provide an intuitive notion such as that of a "partial representation" with a firm logical interpretation, coupled with a clear *computational* grasp of what is involved in parsing and grammaticality judgment, then we can actually assess the consequences of a deficit of this type. Matching these predictions to data in the literature, we find that the deficit hypothesis is apparently supported.

My goals in this discussion will ultimately be more general, however. The computational framework developed to analyze the information-loss deficit can be used to analyze other types of deficits. These models predict different patterns of errors, not found in the agrammatic subjects already alluded to, but found in other types of processing errors. This approach suggests a computational typology of processing deficits which may indicate a more fine-grained analysis of performance errors than is customarily assumed in psychology. In particular, an analysis of the ways in which computational devices can exhibit incorrect behavior strongly indicates that different biases in judgment tasks may signal different types of underlying deficit. Though the kind of analysis pursued in this chapter is very much more abstract than the analysis in Chapter 6, it still yields usable results, in particular in that it indicates a very broad class of deficit theories which cannot yield the pattern of errors seen in agrammatic processing. The possibility that a typology of processing 'faults' (including the results of brain damage, effects of processing overload in normals, or of incomplete learning in children) can be derived based on computational principles is a highly appealing one, particularly within the set of concerns which define current Cognitive Science.

1.4 Connections with Earlier Work

According to Mark Johnson [Johnson 1990], Kaplan and Bresnan were the first to propose a logical description language for linguistic structures.[8] The idea first enters the GB community in work by Marcus and his colleagues [Marcus 1983]. Their interest was in the Determinism Hypothesis, the hypothesis that all, or at least a significant fragment, of natural language pars-

[7] See, for example, [Bradley 1980], [Caplan 1983], [Caplan 1986], [Grodzinsky 1984], [Grodzinsky 1990], [Schwartz 1980], [Zurif 1978], [Zurif 1983]

[8] Cf. [Kaplan 1982].

ing could be done deterministically, so that the parser's representations were indelible, and all its decisions were made once for all. The use of a description language allows for what [Weinberg 1988] dubs a "Minimal Commitment" parsing strategy, where the parser asserts what it is certain of and says nothing about the rest. It is thus deterministic in that once it decides what to do that decision is permanent, but it can postpone such a drastic decision for a certain period of time, waiting for more evidence. I will not be concerned with deterministic parsing in this dissertation, but it is worthwhile to note that the reasons for using logical descriptions are roughly the same. Standard parsing strategies in effect represent local ambiguities which arise in parsing *disjunctively*, either by keeping several fully worked-out structures in play at the same time (a parallel parsing strategy) or by keeping a record of choices made and backtracking when a choice is proven to have been wrong. A minimal commitment parsing strategy deals with local ambiguities by using structural *underspecification*. The result of leaving out bits of structural information is that the parser's working structures are not necessarily trees, an instance of the same problem that arises in trying to extract structure directly from GB-style licensing conditions.[9]

Though the descriptions invoked by Marcus and his descendants are clearly related to the logical description languages descended from Kaplan and Bresnan's work [Kaplan 1982], they have not received nearly the same level of formal, rigorous development. In the realm of feature structures numerous description logics have been proposed, chief among them Kasper-Rounds Logic[10] and Mark Johnson's Attribute-Value Logic.[11] My description language will be a strictly standard first order language, and so resembles most closely that of [Johnson 1990], at least insofar as it deals with features. The structural predicates most closely resemble those of Marcus' D-Theory, though, once again, I will be at some pains to provide them with a strictly formal logic.

The idea of using licensing conditions directly as parsing operations first appears in a paper by Abney & Cole [Abney 1986]. These ideas are developed in Abney's later work [Abney 1987, Abney 1989], but the specification of the grammar underlying these licensing parsers has tended to drift away from any attempt to transparently implement the ideas of theoretical linguists.

[9] Besides [Marcus 1983], these ideas have been developed in [Barton 1985, Berwick 1985, Weinberg 1988]. Marcus et al. refer to these ideas as D-Theory (for "description theory"). Barton & Berwick and Berwick & Weinberg refer to it as Assertion Set Parsing, and Weinberg in her dissertation refers to it, as I mentioned above, as Minimal Commitment parsing.

[10] See [Kasper 1986, Kasper 1987, Moshier 1987, Moshier 1988, Moshier 1991]

[11] See [Johnson 1988, Johnson 1990, Johnson 1991]

In fact neither Abney's work nor that of Marcus and his descendants ever claimed interest in maintaining a transparent relation between the linguists' grammar and the parser's grammar. Yet that is precisely the goal I will strive towards here: to combine these ideas, in effect, in a licensing parser which constructs logical descriptions with the result that we can extract structure more directly from the kinds of principles that theoretical linguists propose.

The application of this approach to aphasia modeling is heavily based on work by Cornell, Fromkin and Mauner.[12] The analysis of agrammatics' processing presented here is also very similar to a suggestion of Linebarger's in [Linebarger 1989]. The rigorous application of the computational model to the analysis of these proposals is new here, as is the extension of these techniques to the general problem of the computational characterization of processing errors.

1.5 Plan of the Dissertation

The dissertation breaks down into three main parts. In the first part (Chapters 2 & 3) I will present the logical underpinnings of the parsing model. In Chapter 2 I give the syntax and semantics of a first order description language $\mathcal{L}$. I will also present a system of axioms for defining tree structures, giving us a way to pick out models which are trees, and to reason about syntactic descriptions based on the assumption that they are trees. Finally I show how sets of ground atomic formulas of $\mathcal{L}$ can be used to pick out particular tree structures. This gives us a way of interpreting partial descriptions as sets of trees, a possibility which will be crucial in my treatment of problems in language breakdown. Then in Chapter 3 I discuss the logic of licensing and licensing conditions, and present a logical model of a grammar interpreter for licensing grammars, which can be thought of as a first pass at a licensing parser.

The second part of the dissertation is concerned with Licensing Grammars and their properties. In Chapter 4 I will begin with some familiar examples, namely context free grammars. I will show how CFGs can be converted straightforwardly into licensing grammars. I will then use those grammars as examples of the working of the parsing model of Chapter 3. We will see that various changes in the parameters which govern the parsing model's operation will yield behavior in the parser very similar to some well-known context-free parsing models. In Chapter 5 I will address the problem of capturing parts of GB theory in the Licensing Grammar formalism. I will discuss the principles and parameters approach to phrase structure, contrasting it with

[12] See especially [Mauner 1990, Cornell 1990, Cornell 1989].

the approach characteristic of context free grammars and their extensions and derivatives. I will thus be concerned to show how $\bar{\mathrm{X}}$ Theory and Θ Theory, in particular, can be implemented in the Licensing Grammar formalism. I will also address the problem of structures derived by movement, that is, in which some kind of non-local syntactic dependencies exist. I will be concerned here to outline the way in which licensing conditions can be used to construct syntactic chains. Note however that I will be restricting my attention to instances of A-movement of maximal projections, ignoring $\bar{\mathrm{A}}$ movement and Head movement, except to note briefly the new problems they present.

In the third part I will present an application of the model developed in the first two parts in an empirical domain, namely the study of language processing after brain damage. In Chapter 6 I will show how some of the licensing conditions developed in Chapter 5 can be extended to capture the very puzzling data concerning the comprehension and acceptability judgments of agrammatic Broca's aphasics. Then, in Chapter 7, I will extend the computational basis of the analysis of Chapter 6 away from the specific problem of modeling parsing in agrammatics and attempt to relate psychological notions of error to computational notions of correctness.

Chapter 2

Description Theory

2.1 $\mathcal{L}$—The Description Language

In this section I present a logical language for making assertions about the structure of trees.[1] This description language (which I will refer to in what follows as $\mathcal{L}$) is a straightforward instance of a first order language with equality, so I will concentrate on presenting its parameters, that is, the non-logical constants, function symbols and predicate symbols. For the rest (connectives, quantifiers, truth values, etc.) I will adopt the standard machinery of first-order logic. The presentation of first order logic follows in essentials that of [Enderton 1972].

We start by assuming a vocabulary of symbols, which we divide up according to their uses in $\mathcal{L}$. First we have the logical symbols: the parentheses, the logical connectives $\rightarrow, \neg, \wedge, \vee$, and $=$, and a set of variables $v_1, v_2, \ldots$, one for each positive integer.

The rest of the symbols which we will use in $\mathcal{L}$ are the non-logical constants, or *parameters* of the language. They include the quantifier symbols $\forall$ and $\exists$, the constant symbols, function symbols, and predicate symbols. For expository reasons, the constants of $\mathcal{L}$ are divided up into several disjoint sets, illustrated in (1). These sets are all assumed to be finite or denumerably infinite. **Value-Names** will be used to denote the values which features can take. **Attribute-Names** will be used to denote feature attribute functions. These are all functions of one argument.[2] The predicate symbols of $\mathcal{L}$ are given in (3), where the elements of P_1 are the 1-place predicates, those

[1] Since I assume that many readers of this dissertation will not be totally familiar with the presentation of a first order logic, I will try to provide footnotes which will explain in more detail concepts which may be presented rather compactly in the text. The specialist reader will thus be able to ignore the footnotes in this section.

[2] I will, however, suggest the usefulness of some two-place functions, in Chapter 5.

of P_2 the 2-place predicates.

(2.1) *Constants of* $\mathcal{L}$

(a) **Node-Names** = $\{n_0, n_1, n_2, \ldots\}$.

(b) **Value-Names** = { *past, plural,* +, ... }

(c) $\perp$, read "undefined".

(2.2) *Functions of* $\mathcal{L}$

(a) **Attribute-Names** = { *tense, number, category,* ... }

(2.3) *Predicates of* $\mathcal{L}$

(a) $P_1 = \{node\}$

(b) $P_2 = \{dom, prec, i\text{-}dom, i\text{-}prec, \approx\}$

The predicates *dom* and *prec* will be used to express dominance and precedence relations, respectively. The two *i*- predicates will be used where it is necessary to enforce *immediate* dominance and precedence.

The reader may have noticed that $\mathcal{L}$ contains two different equality symbols: the (quasi-)logical constant = and the 2-place predicate symbol $\approx$. In fact the logical constant will only be used in providing an axiomatic definition of $\approx$, and it is $\approx$ which we will use in constructing grammatical constraints and logical representations. Looking ahead a little, we will of course interpret = classically. The predicate $\approx$ will behave as a partial, restricted version of =.

In order to name the things whose relationships they describe, expressions of the description language $\mathcal{L}$ make use of constants and variables, and also of complex terms, built up from constants and variables together with function symbols. Formally, we define a collection of term-building operations, one for each function symbol. The term-building operation associated with function f will be called $\mathcal{F}_f$, and, for any expression ϵ,

(2.4) $$\mathcal{F}_f(\epsilon) = f(\epsilon).$$

The set of *terms* is the set of expressions generated from the constants and variables by means of the $\mathcal{F}_f$ operations.[3]

[3] A set is *generated* from another set by some operations if the generated set contains the original set, and is closed under the relevant operations. Technically, the set of terms is the *smallest* such set, that is, the set which contains *only* the constants and variables and complex expressions built from them.

We define an *atomic formula* to be an n-place predicate symbol, together with n terms, which are its arguments. I will write *dom*, *prec* and their immediate variants prefixed to their arguments (as in $dom(x, y)$), but will write $\approx$ infixed between its arguments (as in $x \approx y$).

Given the atomic formulas, we define the more general notion of *well-formed formula (wff)* inductively, as with terms, by defining a collection of formula-building operations. For ϕ and ψ expressions of $\mathcal{L}$:

(2.5) (a) $\mathcal{E}_\neg(\phi) = (\neg\phi)$.

(b) $\mathcal{E}_\rightarrow(\phi, \psi) = (\phi \rightarrow \psi)$.

(c) $\mathcal{E}_\leftrightarrow(\phi, \psi) = (\phi \leftrightarrow \psi)$.

(d) $\mathcal{E}_\wedge(\phi, \psi) = (\phi \wedge \psi)$.

(e) $\mathcal{E}_\vee(\phi, \psi) = (\phi \vee \psi)$.

(f) $\mathcal{Q}^i_\forall(\phi) = \forall v_i\ \phi$.

(g) $\mathcal{Q}^i_\exists(\phi) = \exists v_i\ \phi$.

The set of wffs is the set generated from the atomic formulas by the operations $\mathcal{E}_\neg$, $\mathcal{E}_\rightarrow$, $\mathcal{E}_\leftrightarrow$, $\mathcal{E}_\wedge$, $\mathcal{E}_\vee$, $\mathcal{Q}^i_\forall$, and $\mathcal{Q}^i_\exists$.

In order to avoid cumbersome expressions, parentheses will often be omitted when no structural ambiguity would result.

2.2 Intended Semantics of $\mathcal{L}$

We now take up the problem of how to interpret expressions of $\mathcal{L}$ in terms of the more familiar representations of formal linguistics, namely constituent structure trees. In this section I provide a Tarskian, model-theoretic 'semantics' for the description language $\mathcal{L}$. Keep in mind that the 'meaning' we are interested in in this semantics is not the meaning of linguistic or philosophical semantics. Rather we adopt the mathematical or computer science usage of the term, seeking only to state which trees are 'meant' by a particular set of sentences of $\mathcal{L}$. The sentences of a natural language like English are about entities and concepts of the real world. The sentences of $\mathcal{L}$ are about syntactic structures, and so I will provide a semantics appropriate to that task.

Because we are interested in developing a language for describing trees, the Tarskian approach is useful to us because the notion *describes*, as in a statement like "Δ describes tree $\mathcal{U}$," is closely related to the notion *satisfies a logical formula* or *is a model for a theory.* That is, we will formalize the notion "Δ describes $\mathcal{U}$" as "$\mathcal{U}$ satisfies Δ."

Slightly more formally, we will associate with our description language $\mathcal{L}$ a class of *structures* which give its sentences meaning.[4] A structure is a function $\mathcal{U}$ on the parameters of $\mathcal{L}$, which assigns to the quantifier symbols $\forall$ and $\exists$ a (nonempty) set $|\mathcal{U}|$, the *universe of discourse,* which gives substantive content to notions like "everything" and "something". In our case the universe of discourse includes the nodes and features of a tree, as well as certain other elements, like $\perp$. In what follows, I will refer to these important subsets of $|\mathcal{U}|$ as C, for the set of nodes, and F, for the set of features. Further, $\mathcal{U}$ assigns to each predicate symbol an appropriate relation on the universe of discourse, and to each function symbol an appropriate operation, and to each constant symbol an element in the universe. For an expression ϵ, we write the meaning of ϵ according to a structure $\mathcal{U}$ as $\epsilon^{\mathcal{U}}$.

2.2.1 $\mathcal{L}$ Models

In standard axiomatizations of trees the universe of discourse is partitioned into nodes and labels, and the only function in the system is the labelling function, from nodes to labels. This works well when we are only interested in a single feature of a node (e.g. its category); if we were doing an axiomatization of a standard context-free rewrite grammar this is the kind of model we would want to use. If we are interested in our grammars in more than a single feature of a node, then things get a little more complicated. To capture the kinds of complex symbols in *Aspects*-style syntax [Chomsky 1965], for example, we would need a number of functions in our structures (for category, person and number features, case features, etc.). However, we would still have them just mapping nodes to labels. There would be no compositional interpretation of complex terms in such a structure (where by complex terms I mean here terms of the form $g(f(x))$, $f(f(f(x)))$, etc.).

However, we allow such complex terms in $\mathcal{L}$, and for the following reason. Most theories of grammar within the principles and parameters framework state generalizations over features at various levels of detail. Thus on the one hand we might want a version of $\bar{\text{X}}$-Theory in which a projection and its head must have the same category features (regardless of what those features are). On the other hand, one might also want to state that (in English, say) only elements which are $[-N]$ can assign structural Case under government. We might want to have that under Head-Movement an element takes some of its features with it (including its category, presumably) but leaves others behind (e.g. its θ-assignment features). The conclusion is that the grammar makes reference to features at various levels of granularity. Following a methodology which has proven fruitful in a decade of work in generative phonology, we might conclude (as I will do in this thesis) that a node's features have hierarchical structure, with the hierarchy determined according as rules (principles in this case) make reference to a particular group of features. The intuition is that if a group of features travel together in the eyes of the grammar then those features should at some level be represented as a unit.[5]

We can still maintain the basic division in our structures between nodes and labels. What needs to be redefined is the nature of the mappings allowed as the interpretations of function symbols in $\mathcal{L}$. Given the category features $[\pm N, \pm V]$, if we allow statements such as

$$(N(cat(x)) = +) \wedge (V(cat(x)) = -),$$

and if we simultaneously insist on the neat compositional semantics which is such an appealing feature of first order logic, then it must be the case that $cat(x)$ denotes something, in particular some element of $|\mathcal{U}|$, and that element has to fall within the domain of the function with which we interpret N. So our functions are at least from the nodes and the labels to the labels, not just from the nodes to the labels. If in addition we allow 'features' like "the maximal projection of X" or "the specifier of Y", then functions are at least from nodes $\cup$ labels to nodes $\cup$ labels. We want to have some interpretation of linguists' statements about projections, chains and other syntactic objects which are not necessarily nodes (they are collections of nodes, in the cases mentioned), so there must be objects in our universe of discourse which are the denotata of such statements. When a theory says something about a term like $max(proj(x))$ (where max is for "maximal" and $proj$ is for "projection"), then there must be an object denoted by $proj(x)$. I will treat these syntactic objects as elements of F (the set of feature-label interpretations in $|\mathcal{U}|$), though we do not normally think of them as features. The alternative is just to define another partition in $\mathcal{U}$.

In addition to nodes and labels, we have a special element '$\perp$', which is the 'dead state', the value of functions where they would otherwise be undefined. (I will allow this symbol to denote itself, since no confusion should arise from

[4] The notion of "structure" is given a certain formal definition in mathematics, as a set of objects together with some relations and operations on elements of that set. The potential confusion engendered by the linguistic usage of the term will actually be useful to us, since I will be formalizing syntactic structures as structures in this mathematical, algebraic sense.

[5] Though the representational system I adopt, with structured node labels, is formally very similar to the representations of the various feature-structure- and unification-based grammars, the motivation for adopting a particular feature is, in this system, strictly derived from the form of the grammar, and not vice versa. Whether or not this distinction is significant is not absolutely clear to me at this point.

this. If in some theory we have a sentence like $f(x) = \perp$, and if $\mathcal{U}$ satisfies that theory, then it will always have to be true that $f(x)^{\mathcal{U}} = \perp^{\mathcal{U}} = \perp$ in $\mathcal{U}$.)

$$|\mathcal{U}| = C \cup F \cup \{\perp\}. \tag{2.6}$$

The elements of the universe stand in certain relations which we can characterize as: "dominates" or "has as a constituent" (**D**), and "precedes" (**P**).[6] I will define these relations to be strict partial orders on subsets of the domain of discourse (in particular we are only interested in them as they relate elements of C).[7]

Structures are technically speaking functions, from elements of the syntax to elements of the universe, or relations or operations on elements of the universe. However, it is often useful to characterize them as tuples, where we give the relations and functions which will actually be important as elements of the tuple. Accordingly I define *labeled, cross-ordered structures* (LCOs), which express those aspects of our intended models which are of particular interest. For example, a structure for $\mathcal{L}$ must define the relation $i\text{-}dom^{\mathcal{U}}$. However, when I actually define the conditions under which an immediate dominance condition is satisfied by some structure, I will only make reference to the dominance partial order **D**. Therefore, we can ignore the interpretation of *i-dom* in $\mathcal{U}$ in our discussions, even though in a classical First Order Logic there must be a relation $i\text{-}dom^{\mathcal{U}}$.

Def'n 1 *A* labeled, cross-ordered structure *(LCO) is a 4-tuple*

$$\langle |\mathcal{U}|, \mathbf{D}, \mathbf{P}, \{f_i\} \rangle,$$

where i may be taken to range over the elements of **Attribute-Names**.

Below (§2.3) I will define tree structures in terms of LCO's which meet certain additional conditions. However, it will prove more useful to define the satisfaction relation in these more general terms. Before defining satisfaction, however, I will present an example of the kind of model I have in mind, since

[6] Thus $\langle \alpha, \beta \rangle \in \mathbf{D}$ means that β is a constituent of α. Relations are sometimes written infixed, e.g. $\alpha\mathbf{D}\beta$, and sometimes written set theoretically. We could say either:

$$\mathbf{D} = \{\langle \alpha, \beta \rangle : \alpha\mathbf{D}\beta\}$$

or equivalently:

$$\alpha\mathbf{D}\beta \text{ iff } \langle \alpha, \beta \rangle \in \mathbf{D}.$$

I will generally use the set-of-pairs notation.

[7] Often dominance is treated as a weak partial order, rather than a strict one. The choice is somewhat arbitrary, and makes some things easier to express, some things harder.

the treatment of features which I will offer is not as intuitive or familiar as the treatment of constituency. Suppose we start with a more or less conventional tree, depicting (very roughly!) the kind of structure which many current versions of GB propose for the verb phrase at d-structure. A traditional sort of depiction of this structure is given in Figure 2.1. (The annotation under the V node is a fairly standard representation of a thematic grid. The underlined grid position is the external argument role. Thematic role assignment is represented by the coindexing of grid positions with their associated argument NPs.)

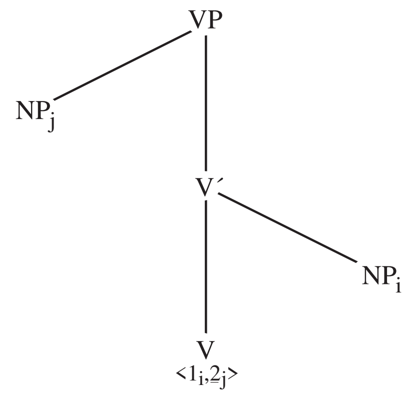

Figure 2.1: A d-structure verb phrase.

Convention plays a very large role in the way we interpret such pictures, as becomes (somewhat painfully) obvious when we try to characterize logically what such depictions are supposed to mean. In the rest of this section I will try to unpack the tree in Fig. 2.1 into an LCO. This discussion will serve to make clearer the kinds of objects, relations and functions which I will be assuming as interpretations of statements and theories in $\mathcal{L}$.

In the illustrations which accompany the ensuing discussion (e.g., Fig. 2.2) I will represent the elements of $|\mathcal{U}|$ as small circles, as if they were the nodes of a graph. The dominance and precedence relations are given as is usual in pictures of trees (cf. Fig. 2.1), but I have in addition made the arcs representing (immediate) dominance relations heavier than the arcs which represent the functions, and filled in the circles representing elements of C. Since the main interest here is in the functions and the elements they relate, I have not tried to give a perfectly explicit depiction of the dominance and precedence relations. Rather, I will leave the structural skeleton of the tree depicted as in Fig. 2.1, but the reader should understand that the heavy arcs that represent the dominance relation are directed (that is, they should all have arrowheads on their south ends), and the full dominance relation is actually the transitive closure of the arcs depicted here.

Where elements of the universe are denoted by constant symbols I have written the constant symbol in 12-point type, next to the element which that symbol denotes. Function symbols have been set in smaller type along the arcs which define them. For readability I have written, e.g., *cat* rather than f_{cat}, a device I will employ throughout this dissertation. Arcs not represented in any of these illustrations are presumed to lead to the dead-state $\perp$. Note that in these simple illustrations I make no claims to deep grammatical insights or compatibility with any existing grammatical proposals.

First I present the basic skeletal tree, in Figure 2.2. Since the category-features standardly used to label the nodes in a tree are here represented as values of features, they are not represented in this preliminary illustration. Rather, the nodes are labeled with the names of the constants which denote nodes in the tree. Thus the VP node of Fig. 2.1 becomes the denotation of the term n_1 in Fig. 2.2, the "subject" NP becomes n_2, etc.

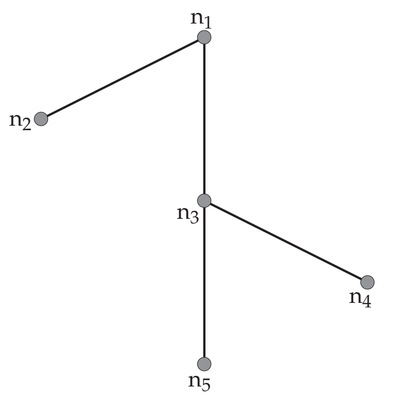

Figure 2.2: Bare tree, with node-names.

Such a structure could be a model for a theory including the sentence in (2.7), for example.

(2.7) $$node(n_1) \wedge node(n_2) \wedge node(n_4) \wedge$$
$$dom(n_1, n_2) \wedge dom(n_1, n_4) \wedge prec(n_2, n_4).$$

Such a sentence does not, of course, completely determine the structural properties of the tree in Figure 2.2. Node n_3 is not even mentioned, for

example. Nonetheless the tree in Figure 2.2 does satisfy the expression in (2.7). Such an expression thus serves as a *partial description* of the tree in Figure 2.2, a concept which will concern us a good deal later in this chapter.

In Figure 2.3 I present the same basic tree together with the functions which give the category of each node. The so-called "lexical categories" N, V, Adj and P are normally assumed to be characterized by feature matrices given schematically as [±N, ±V]. I have renamed these attributes here *noun*, for "nominal", and *verb*, for "verbal". I have also assumed that the category features are part of a larger family of features, essentially those which are shared among the projections of a single head category. I have given this feature family as the value of the function *core*.

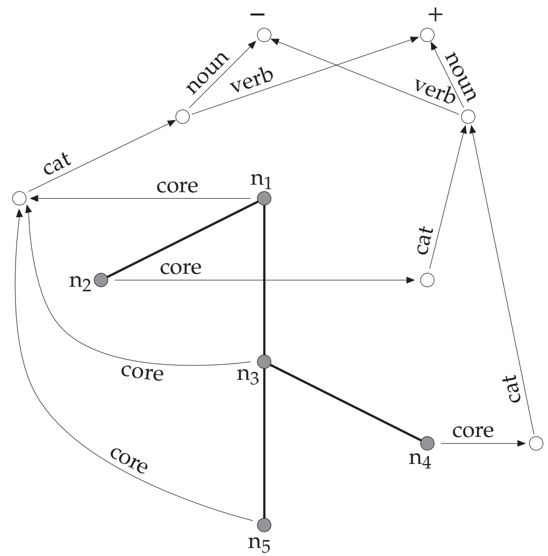

Figure 2.3: A sample category system.

I have illustrated in Figure 2.3 an interesting property of $\mathcal{L}$, which I will take advantage of in later chapters. As mentioned above, we can think of n_1 as VP, n_2 and n_4 as the external and internal argument NPs, n_3 as V′, and n_5 as V^0. Accordingly the category of n_2 and n_4 is [+N,−V], and the category of n_1, n_3 and n_5 is [−N,+V]. We can think of the individual denoted by $cat(core(n_1))$, among other expressions, as the category Verb, and the individual denoted by $cat(core(n_2))$ as the category Noun. Note that the category of n_2 and n_4 is the same, but their core features are different. In contrast, the function f_{core} maps n_1, n_3 and n_5 to a single individual. In such a structure the following are some of the true statements:

$$cat(core(n_2)) = cat(core(n_4)), \tag{2.8}$$
$$core(n_1) = core(n_3). \tag{2.9}$$

Furthermore it evidently follows from (2.9) that

$$cat(core(n_1)) = cat(core(n_3)).$$

But it is not the case that from (2.8) it follows that

$$core(n_2) = core(n_4).$$

This use of identity of features is a potentially useful implementation of feature-percolation, as might happen in accord with the conditions of $\bar{\mathrm{X}}$ Theory. Passing features up the tree amounts to equating the values of certain features of successive nodes. Some beneficial results are that information is propagated in the projection directly: if we assert that the maximal projection has certain thematic role or Case features, then, if these are core features, they automatically characterize the head as well, and all the intermediate projections, without further effort. This general mechanism of "feature-assignment by feature-identity" is often exploited in unification-based grammars.

The sharing of features also offers a way to talk about projections as objects, since there is an individual in the universe of discourse which effectively stands for a projection. Looked at the other way around, we could define an $\bar{\mathrm{X}}$ projection as the set of all nodes x for whom f_{core} is the same. Keep in mind that this is not at all the same as saying that an $\bar{\mathrm{X}}$ projection is a set of nodes whose category features, Case features, etc. happen to be the same—identity of core features is a much stronger requirement. This may allow us to avoid using explicit projection indices (cf. [Speas 1991])—the shared core feature-set in a sense *is* the projection index.

The final feature-system which I will illustrate is the thematic system. Figure 2.4 illustrates the functions which might be used to interpret the results of θ-role assignment in a GB grammar.

Following standard presentations of Theta Theory (e.g., [Stowell 1981]), a thematic grid is taken to be a collection of *slots* or grid *positions*. Here, in Fig. 2.4, I have treated grid positions as if they formed a sort of sequence, tied together by a function which I have suggestively labeled s, to recall the successor function of number theory and axiomatic arithmetic. Thus, for a node X, the term $arg1(X)$ gives its first grid position and $s(arg1(X))$ gives

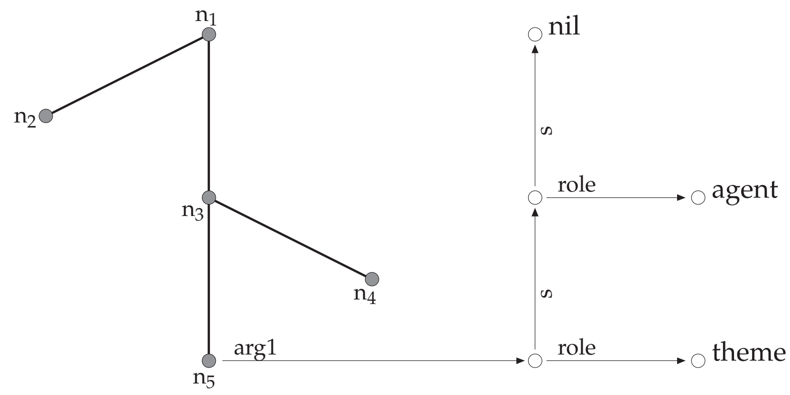

Figure 2.4: A sample thematic grid.

the next one and so forth. Normally we think of the θ-grid as an unordered set of θ-roles, or, more generally, of argument slots (cf. [Stowell 1981], who notes that a great deal more than just thematic information may be tied to grid slots). On the other hand the actual projection of θ-roles into the syntax is subject to some sort of thematic hierarchy. Either the hierarchy defines a function which selects the next role to assign from an unordered set or it imposes an order on the set. This latter option seems simpler so I have used it in this example.

In this example I have assumed a system which passes the undischarged portion of a node's θ-grid up to its mother. If each node assigns the first undischarged node in the grid that it inherits (from its daughter or from the lexicon), then each successive θ-role in the grid will be assigned by each successive ancestor of the head to its sister until all of the grid positions have been discharged, in accordance with Θ Theory.[8] (I follow here suggestions in, e.g., [Koopman 1988, Sportiche 1990] that θ-role assignment should take place under strict sisterhood.)

Accordingly terms of the form $arg1(X)$ are interpreted as denoting the portion of a θ-grid inherited by node X. Thus $arg1(X)$ gives the role actually assigned by X and $s(arg1(X))$ gives the portion of the grid inherited by X's mother. Passing the undischarged grid up the tree from a θ-assigner X to

[8] I have not in these simple examples bothered to distinguish external θ-role assignment from the assignment of internal roles, though I have somewhat tendentiously drawn the "external" argument, n_2, on the left and the "internal" argument, n_4, on the right.

its mother Y is simply a matter of asserting:

$$arg1(Y) \approx s(arg1(X)).$$

And θ-role assignment from one node X to another Z is simply a matter of asserting something like:

$$arg1(X) \approx core(Z).$$

Figure 2.5 shows the structure discussed above after the object NP, n_4, has been identified with the first grid position of the V node, n_5. In the process the remainder of the grid has been assigned to the first projection of the V, n_3.

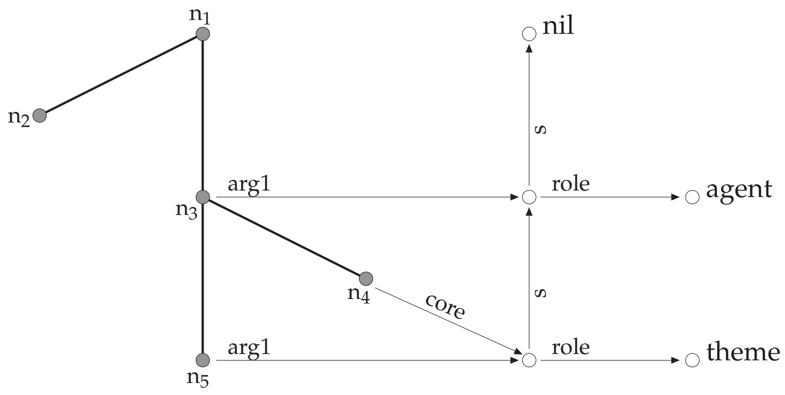

Figure 2.5: Assignment of the Theme role.

The particular variant I have given (essentially following [Stowell 1981] in *identifying* the argument with its associated grid position, rather than merely *copying* a thematic role feature onto the NP) has the potential advantage that any information associated with a grid position (including any category-selection features, or inherent Case features, or semantic selection features) will be inherited by the assignee, and the attempt to associate an XP with a particular argument position will be blocked if that XP's features clash with the features specified in the grid. For example, the attempt to assign a PP to a position selecting an NP would introduce a contradiction in that the *noun* feature would receive both '+' and '−' values.

The next figure, Figure 2.6, shows how the process proceeds. The second position in the verb's grid is now the first, exposed position in the V′'s grid,

and it is associated with the external argument NP, n_2. Its mother node, n_1, inherits the remainder of the grid, in this case *nil*, which marks the end of the grid, so that n_1 is marked as a saturated predicate.

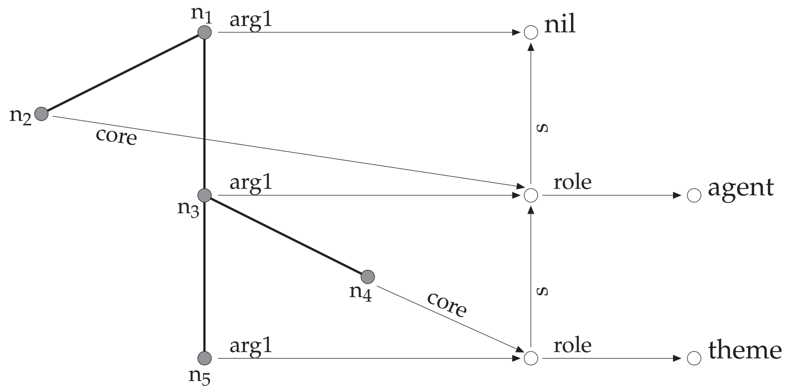

Figure 2.6: Assignment of the Agent role.

Note that although θ-role assignment in a system like this is most easily discussed as if it involved the passing of the undischarged portion of the grid on up the tree, in fact there is only one copy of the θ-grid in this structure. Technically what happens is more like projections of the head inheriting pointers into the head's θ-grid rather than copies of the grid's remainder.

It is somewhat non-standard to take so much time illustrating the models which are assumed for a logical language. Normally we use logic to relate models about which we already have strong intuitions to expressions in a formal system which allows us to do rigorous proofs which preserve those intuitions. To some extent this holds here: we understand tree structures well enough to trust our intuitions in deciding whether x does or does not dominate y in a given tree and so forth. However, in principles and parameters theories of grammar the uses of features have at this point gotten fairly far ahead of generally shared assumptions about their nature, so it is worth while to take some time to understand the models I have in mind for expressions of $\mathcal{L}$. Still, there are many things about features in a grammar which are fairly familiar, and about which there is considerable agreement at an intuitive level. Thus we would probably all agree that, if the role associated with the second grid position of X is **agent**, and the role to be assigned by Y to its sister Z is the second role in X's grid, then it should be a valid conclusion that Z gets the agent role. One way to express this argument in

$\mathcal{L}$ is with the following sentences.

$$\begin{array}{lrcl}
(2.10) & role(s(arg1(X))) & = & agent \\
& arg1(Y) & = & s(arg1(X)) \\
& core(Z) & = & arg1(Y) \\
\hline
& role(core(Z)) & = & agent
\end{array}$$

It should be clear (perhaps with a little tracing of lines) that this argument is verified by the structure depicted in Fig. 2.6 if we take X to be n_5, Y to be n_3 and Z to be n_4. Thus, although structures like the one I have illustrated seem a little outlandish compared to the illustrations in a typical linguistics journal article, nonetheless these structures do in fact preserve the validity of arguments which we know we want to be able to make in our theory of the grammar.

2.2.2 Satisfaction and Truth

Returning to the definition of the semantics of $\mathcal{L}$, we want to be able to say more formally, given a sentence ϕ of $\mathcal{L}$ and a structure $\mathcal{U}$, that ϕ is a true statement about $\mathcal{U}$ (written $\models_{\mathcal{U}} \phi$). We will see that this notion, of "truth in a structure", lines up perfectly with our desire to know whether or not ϕ *describes* $\mathcal{U}$. The notation $\models_{\mathcal{U}} \phi$ is also often read "$\mathcal{U}$ *satisfies* or *models* ϕ."

Note that in our definition of structure no provision was made for variables and their denotations. In order to define the *satisfies* relation ($\models$), we need a *variable assignment function*—let us call it s—from the set of variables to $|\mathcal{U}|$. We then say "$\mathcal{U}$ satisfies ϕ with s," writing $\models_{\mathcal{U}} \phi\,[s]$.

In order to define the denotations of terms, we extend s (which properly applies only to variables) to a function $\overline{s}$, from terms in general to $|\mathcal{U}|$.

(2.11) *Denotation of Terms*

(a) For v a variable, $\overline{s}(v) = s(v)$.

(b) For c a constant symbol, $\overline{s}(c) = c^{\mathcal{U}}$.

(c) For f a function symbol and t a term, $\overline{s}(f(t)) = f^{\mathcal{U}}(\overline{s}(t))$.

We now define "satisfies with s" for atomic formulas.

(2.12) *Satisfaction of Atomic Formulas*

For terms t, t_1, t_2:

(a) $\models_{\mathcal{U}} node(t)\,[s]$ iff $\overline{s}(t) \in C$.

(b) $\models_{\mathcal{U}} dom(t_1, t_2)\,[s]$ iff $\langle \overline{s}(t_1), \overline{s}(t_2)\rangle \in \mathbf{D}$.

(c) $\models_{\mathcal{U}} prec(t_1, t_2)\,[s]$ iff $\langle \overline{s}(t_1), \overline{s}(t_2)\rangle \in \mathbf{P}$.

(d) $\models_{\mathcal{U}} (t_1 = t_2)\,[s]$ iff $\overline{s}(t_1) = \overline{s}(t_2)$.

(e) $\models_{\mathcal{U}} (t_1 \approx t_2)\,[s]$ iff $\overline{s}(t_1) = \overline{s}(t_2) \neq \perp$.

(f) $\models_{\mathcal{U}} i\text{-}\pi(t_1, t_2)\,[s]$ iff $\langle \overline{s}(t_1), \overline{s}(t_2)\rangle \in \Pi$ (for π either *dom* or *prec* and Π either $\mathbf{D}$ or $\mathbf{P}$, respectively), and there is no $\alpha \in |\mathcal{U}|$ such that $\langle t_1, \alpha\rangle \in \Pi$ and $\langle \alpha, t_2\rangle \in \Pi$.

The logical connectives and quantifiers receive their usual interpretations, as in the inductive definition given in (13).

(2.13) *Satisfaction of Non-Atomic Wffs*

For formulas ϕ, ψ, variables v:

(a) $\models_{\mathcal{U}} \neg\phi\,[s]$ iff $\not\models_{\mathcal{U}} \phi\,[s]$.

(b) $\models_{\mathcal{U}} (\phi \to \psi)\,[s]$ iff $\not\models_{\mathcal{U}} \phi\,[s]$ or $\models_{\mathcal{U}} \psi\,[s]$.

(c) $\models_{\mathcal{U}} (\phi \leftrightarrow \psi)\,[s]$ iff $\models_{\mathcal{U}} (\phi \to \psi)\,[s]$ and $\models_{\mathcal{U}} (\psi \to \phi)\,[s]$.

(d) $\models_{\mathcal{U}} (\phi \wedge \psi)\,[s]$ iff $\models_{\mathcal{U}} \phi\,[s]$ and $\models_{\mathcal{U}} \psi\,[s]$.

(e) $\models_{\mathcal{U}} (\phi \vee \psi)\,[s]$ iff $\models_{\mathcal{U}} \phi\,[s]$ or $\models_{\mathcal{U}} \psi\,[s]$.

(f) $\models_{\mathcal{U}} \forall v\, \phi\,[s]$ iff for every $d \in |\mathcal{U}|$, we have $\models_{\mathcal{U}} \phi\,[s(v|d)]$.

(g) $\models_{\mathcal{U}} \exists v\, \phi\,[s]$ iff there is some $d \in |\mathcal{U}|$ such that $\models_{\mathcal{U}} \phi\,[s(v|d)]$.

The notation $s(v|d)$ is taken to mean a function identical to s, except at v, where it takes the value d. We can think of this as scanning all the possible assignments to v while holding all other assignments constant.

If, for an expression ϕ of $\mathcal{L}$, there are no free variables in ϕ, then the choice of s is irrelevant, and we can just say $\models_{\mathcal{U}} \phi$. If $\mathcal{U}$ satisfies ϕ for any choice of s, we say $\mathcal{U}$ is a *model* for ϕ.

2.3 $\mathcal{T}$—A First-Order Theory of Trees

In this section I will outline an axiomatization and proof system for a logical theory of tree-structures. Such a system could serve any of a number of purposes, and it is important therefore to keep in mind the particular purpose I intend here. We could attempt here to present an axiomatic *definition* of tree structures (cf. [Rodman 1977, Partee 1990]).[9] In that case we would primarily be interested in developing a system of axioms $\mathcal{A}$ in such a way that the models for $\mathcal{A}$ were exactly the tree structures. This is quite difficult to do, especially if we want to insist that tree structures be of arbitrary but finite size. Here, however, the ultimate goal is not a mathematical characterization of syntactic structures, but a parsing model which can reason about such objects. Therefore we also want a set of axioms to use as the basis of a deductive calculus, which in turn defines a significant part of the parsing model's operation. Suppose, for example, that the parser has constructed a theory Δ, consisting of some sentences of $\mathcal{L}$ which describe the syntactic structure of its input. There are implicit in Δ very many facts about that syntactic structure, some of which we might want to be able to make explicit; we need a way of drawing at least some of the possible inferences from sets of facts about syntactic structure. For this we need a proof system, which in its own turn depends on a system of axioms. As an example, if we have a theory Δ among whose terms are nodes n_1, n_2 and n_3, and Δ tells us that $dom(n_1, n_2)$ and $dom(n_2, n_3)$, but nothing has been explicitly stated about the relation between n_1 and n_3, we would certainly want to be able to conclude that $dom(n_1, n_3)$ was true in any model satisfying Δ. Accordingly we need an axiom which states that the dominance relation is transitive. On the other hand there are vastly many hugely complex tautologies which are undoubtedly true in any model satisfying Δ, but which add nothing of syntactic interest to our explicit description of the input sentence's syntax. There are even facts specifically about trees which are irrelevant to the parser's operation. For example, it is unnecessary for the parser ever to conclude that its representation is finite—the finiteness of the input, together with a carefully constructed grammar should ensure that this is so, and if it is not so, the parser will nonetheless be unable to use the knowledge that its representation must not become infinite in size in any constructive way.

The axiom system has two parts (not including the grammar, which will be taken up in the ensuing chapters). On the one hand we have the logical axioms, which I will refer to collectively as Λ, which determine the properties of the logical connectives and regulate their interactions, providing us with a familiar background logic. On the other hand we have the tree axioms $\mathcal{T}$. These define for us the properties peculiar to the parameters of $\mathcal{L}$ and regulate their interactions in such a way that $\mathcal{T}$ is satisfied by all objects which agree with our intuitive understanding of tree structures. Put another

[9] In this dissertation I will assume that syntactic structures are trees. The phrase markers of [Chomsky 1955/75] or those of [Lasnik 1977] are somewhat different from trees in a number of details. However, as the illustrations accompanying most GB journal articles demonstrate, in practice standard tree structures are generally assumed.

way, the axioms of $\mathcal{T}$ express the things that we can validly conclude from a body of facts once we know that they are facts about a tree.

2.3.1 Logical Axioms

Once again the presentation follows [Enderton 1972] in all significant details. This system differs from most similar proof systems for first order logic in allowing all tautologies of sentential logic as axioms, rather than starting with a small set from which the other tautologies can be derived. Doing it this way makes formal deductions much simpler. Note that these "axioms" are actually axiom *schemata*, from which an infinite set of axioms can be derived by uniformly substituting formulas of $\mathcal{L}$ for the template variables ϕ and ψ.[10] In these schemata we use not only the sentence place-holders ϕ and ψ but also formal variables x and y, which stand for any actual variable (i.e. element of the set $\{v_1, v_2, \ldots\}$), and t, t_1, t_2, which stand for arbitrary terms.

(2.14) *Logical Axioms* (Λ)

- Λ1. The propositional tautologies are all axioms of Λ.[11]
- Λ2. $\forall x\, \phi \to \phi^x_t$, where t is substitutable for x in ϕ. ("Universal Instantiation")
- Λ3. $\forall x\, (\phi \to \psi) \to (\forall x\, \phi \to \forall x\, \psi)$.
- Λ4. $\phi \to \forall x\, \phi$, where x does not occur free in ϕ.

(2.15) *Equality Axioms of* Λ

- Λ5. $\forall x\, x = x$.
- Λ6. $\forall x \forall y\, (x = y) \to (\phi \to \phi')$ where ϕ is atomic and ϕ' is the result of replacing zero or more occurrences of x in ϕ with y. ("Substitution of Equals for Equals")

[10] Axioms which instantiate these schemata are all formulas of $\mathcal{L}$, and so part of a formal, syntactic calculus for (ultimately automatically) reasoning about syntactic structure. Statements like "where x does not occur free in ϕ" are, of course, not expressions of $\mathcal{L}$. But because these are axiom *schemata* they do not have to be—such statements regulate how the schema can be instantiated, but once instantiated, it is a pure expression in $\mathcal{L}$.

[11] The propositional tautologies are sentences which are true no matter what the truth values of their constituent sentences. Thus $P \vee \neg P$ is a tautology. It is an axiom of first order logic in that it is likewise true no matter what closed sentence of the logic we substitute for P.

The phrase "substitutable for v in ϕ" is present in Λ142 to prevent instances of the axiom scheme in which a variable in t inadvertently becomes bound by a quantifier in ϕ. The problem arises because, in a strictly formal system, it is very hard to legislate a prudent choice of variable names in ϕ and t. The key to the definition (which is given in [Enderton 1972, p.105]) is that, given a formula $\forall x\ \phi$, a term t, and a variable y which may occur in ϕ, t is substitutable for y in $\forall x\ \phi$ just in case either (a) y does not occur free in $\forall x\ \phi$ or (b) x does not occur at all in t and t is substitutable for y in ϕ. This has the effect of recursively examining all the quantifiers in a formula to verify that none of them will accidentally bind anything in t.

In addition to the axioms in (14) and (15) we also need the following, which allow us to translate back and forth among the various connectives. [12]

(2.16) *Additional Axioms of* Λ

Λ7. $(\phi \to \psi) \leftrightarrow (\neg\phi \vee \psi)$.

Λ8. $\neg(\phi \wedge \psi) \leftrightarrow (\neg\phi \vee \neg\psi)$.

Λ9. $(\phi \leftrightarrow \psi) \leftrightarrow ((\phi \to \psi) \wedge (\psi \to \phi))$.

Λ10. $\exists x\ \phi \leftrightarrow \neg\forall x\ \neg\phi$.

The sole rule of inference in this system is *modus ponens*. That is, given as premises both $(\phi \to \psi)$ and ϕ, we are allowed to conclude ψ. Given a set of $\mathcal{L}$-formulas Δ, a *theorem* of Δ is a member of the set generated from Δ by modus ponens. A *deduction* of ϕ from Δ is a sequence of formulas $a_1, a_2, \ldots, a_n$ such that $a_n = \phi$ and every a_i in the sequence is either in Δ or is obtained by modus ponens from formulas occurring earlier in the sequence. These two notions, *theorem* and *deduction*, are closely related in the obvious way: ϕ is a theorem of Δ just in case there is a deduction of ϕ from Δ. We can read "$\Delta \vdash \phi$" either as "ϕ is a theorem of Δ," or "ϕ is deducible from Δ." Given the axioms of this section, this deductive system is sound and complete.

[12] In standard presentations of first order logic (like [Enderton 1972]) it is common to introduce only two connectives (the most common choices are $\neg$ and $\to$, or $\neg$ and $\vee$), and one quantifier symbol ($\forall$), and to define the other symbols as strictly abbreviatory conventions. This makes proving metatheorems about first order logic much simpler. One need only give a semantics for the primitive symbols, and in proofs which proceed by cases the number of cases is drastically reduced. However it is not my goal to prove anything new about first order logic, but rather to use it to express the logical foundations of a parsing model. Including the full complement of connectives into $\mathcal{L}$ makes it much more concise, with much less cumbersome expressions. This is where I have to "pay the piper" for that expressiveness, providing as axioms of Λ relations among the connectives that would normally be given as definitions of abbreviatory conventions.

2.3.2 Equality in the Axioms of $\mathcal{T}$

There are several restrictions which we need to put on what can equal what in this system, in order to capture what we mean by the features of nodes, feature-attributes and feature-values. As a first example, consider the following problem. Suppose we were able to construct a deduction in which the final formula was

(2.17) $$\textit{singular} \approx \textit{plural}.$$

Clearly this is wrong. In fact it is always wrong to equate any member of **Value-Names** with any other member. For these elements *syntactic equality* holds. That is, two elements are equal just in case they are formally identical, as in

(2.18) $$\textit{singular} \approx \textit{singular}.$$

We implement this restriction in the following axiom scheme.

(2.19) *Distinctness of Feature Values*

$\mathcal{T}$1. $c_1 \neq c_2$, where c_1 and c_2 are distinct value-names.

Normally a logical system will either be restricted to syntactic equality ("first-order logic without equality") or it will allow it everywhere ("first-order logic with equality"). This system is a hybrid, which makes necessary the extra pseudo-equality relation $\approx$, as well as extra axioms of equality like $\mathcal{T}$191. Note that the problems which arise in this section also arise in the feature-structure logics, and my solutions follow theirs in most details (in particular the presentation in [Johnson 1990], but cf. also [Moshier 1988]).

Another problem with equality arises also from a sense in which this system is slightly non-standard. According to our definition of terms, alongside such well-behaved terms as those in

(2.20) $$\textit{person}(n) \approx \textit{3rd},$$

we have such ill-behaved examples as the first term in

(2.21) $$\textit{person}(\textit{3rd}) \approx x.$$

The problem is that, in order to be a standard first order logic, and inherit all the nice properties which they have, the functions of this system have to be total, that is, defined for all elements of $|\mathcal{U}|$. In fact, however, the feature attributes ought to be partial functions. In our example, the function denoted by *person* ought not to be defined for *3rd*, or any other member of **Value-Names**, for that matter. We worked around this problem by introducing a special element into $\mathcal{L}$, the dead state symbol $\perp$. Using this, we can simulate partial functions on the domain of discourse by assigning them, where they would otherwise be undefined, the value $\perp$. Thus, in the case of the bad example in (2.21), we would have

$$\textit{person}(\textit{3rd}) = \perp.$$

In order to maintain the soundness of our proof system, it is necessary that we not be able to derive statements like (2.21) without explicitly recognizing them as contradictions, that is, instances of $\phi \wedge \neg\phi$. Accordingly, we introduce the following axiom schema.

(2.22) *Values have no attributes:*

$\mathcal{T}2$. $f(c) = \perp$ for any $f \in$ **Attribute-Names** and $c \in$ **Value-Names** $\cup \{\perp\}$.

We now introduce $\approx$ formally and define its behavior in terms of $=$.

(2.23) *Definition of* $\approx$

$\mathcal{T}3$. For all terms t_1, t_2,

$$(t_1 \approx t_2) \leftrightarrow (t_1 = t_2 \wedge t_1 \neq \perp).$$

This restricts the use of $\approx$ to terms whose values are defined. Note that all the properties of $=$ hold of $\approx$ except for reflexivity (that is, $\approx$ is symmetric, transitive, and substitution of $\approx$ for $\approx$ preserves truth). As for reflexivity, note that $\perp \not\approx \perp$ for instance. In general:

(2.24) $$t_1 \approx \perp \rightarrow t_1 \not\approx \perp,$$

for all terms t_1.

When, in the course of reasoning about some natural language input, we derive a statement like

(2.25) $$f(g(t_1)) \approx t_2$$

for functions f and g and terms t_1 and t_2, we want to be certain that the subterm $g(t_1)$ is defined. Because of the nature of the $\approx$ relation we can say "$g(t_1)$ is defined" in the following way.

(2.26) $$g(t_1) \approx g(t_1)$$

Given the reflexivity of standard $=$, the only way (2.26) could be false is if $g(t_1) = \perp$. If $g(t_1) = \perp$ then so does $f(g(t_1))$, by $\mathcal{T}222$. Accordingly we have the following.

(2.27) *Sub-terms*

$\mathcal{T}4$. $\forall t\ f(t) \approx x \rightarrow t \approx t$.

2.3.3 Dominance & Precedence

We now turn to the predicates of $\mathcal{L}$, and define for them the properties which our model theory assumes.

First, dominance and precedence are modeled by the (strict) partial orders **D** and **P**, so in syntactic deductions they must obey the axioms for strict partial orders, given below in (28).[13]

(2.28) *Strict Partial Order Axioms*

$\mathcal{T}5$. *Irreflexivity Axioms*

i. $\forall v_1 \neg \textit{dom}(v_1, v_1)$.

ii. $\forall v_1 \neg \textit{prec}(v_1, v_1)$.

$\mathcal{T}6$. *Transitivity*

i. $\forall v_1 \forall v_2 \forall v_3\ \textit{dom}(v_1, v_2) \wedge \textit{dom}(v_2, v_3) \rightarrow \textit{dom}(v_1, v_3)$.

ii. $\forall v_1 \forall v_2 \forall v_3\ \textit{prec}(v_1, v_2) \wedge \textit{prec}(v_2, v_3) \rightarrow \textit{prec}(v_1, v_3)$.

$\mathcal{T}7$. *Immediacy*

i. $\forall v_1 \forall v_2\ \textit{i-dom}(v_1, v_2) \leftrightarrow (\textit{dom}(v_1, v_2) \wedge \neg\exists v_3\ (\textit{dom}(v_1, v_3) \wedge \textit{dom}(v_3, v_2)))$.

ii. $\forall v_1 \forall v_2\ \textit{i-prec}(v_1, v_2) \leftrightarrow (\textit{prec}(v_1, v_2) \wedge \neg\exists v_3\ (\textit{prec}(v_1, v_3) \wedge \textit{prec}(v_3, v_2)))$.

In addition to the partial order axioms Irreflexivity ($\mathcal{T}285$) and Transitivity ($\mathcal{T}286$), I have provided here the axioms governing the behavior of the two immediate ordering predicates. Note that the first consequence of an immediate ordering statement like $\textit{i-dom}(a, b)$ is the assertion of the plain vanilla ordering relation $\textit{dom}(a, b)$.

2.3.4 Constituency

There are many structures which are partial orders but not constituent structures. We need to express what we mean by constituency. It can fairly easily

[13] Familiar examples of strict partial orders include the less-than relation ($<$) and the strict subset relation ($\subset$). These are contrasted with the weak partial orders less-than-or-equal-to ($\leq$) and subset ($\subseteq$), where instead of irreflexivity, we substitute an axiom of reflexivity ($x\mathbf{R}x$ is always true for a weak order relation $\mathbf{R}$), and we add an axiom of antisymmetry ($x\mathbf{R}y \wedge y\mathbf{R}x \rightarrow x = y$). Transitivity holds in weak partial orders just as in strict ones.

Note that in characterizing strict partial orders we do not need an axiom corresponding to the antisymmetry of weak orders. The corresponding property (asymmetry: $\neg(\textit{dom}(x, y) \wedge \textit{dom}(y, x))$) follows as a theorem by applying transitivity and then irreflexivity. See below.

be shown that the following axioms give us just the properties which we normally take to define constituency. In the statement of these axioms I will employ a useful abbreviation: $nodes(v_1, \ldots, v_n)$ will be taken to abbreviate $node(v_1) \wedge \ldots \wedge node(v_n)$. I will also make use of a similar abbreviation for multiple instances of $v_1 \not\approx v_2$: I will write $distinct(v_1, \ldots, v_n)$ for the application of '$\not\approx$' to the cross product of $\{v_1, \ldots, v_n\}$, less the reflexive pairs $\langle v_1, v_1 \rangle$, etc.[14]

The three axioms which implement our standard notions of constituent structure are given in (29), below. First, every tree is *rooted* ($\mathcal{T}$298), that is, there is one constituent which contains all the others. Second, and somewhat less obvious, every node of the tree must be ordered in at least one way with respect to every other node ($\mathcal{T}$299). Finally, each subconstituent will inherit the precedence properties of the constituents which contain it ($\mathcal{T}$2910). This is the axiom which guarantees the integrity of constituents, as we will see when we proceed to unpack the content of these axioms in some theorems about trees.

(2.29) *Axioms of Constituency*

$\mathcal{T}$8. *Rootedness*

$$\exists v_1 \; node(v_1) \wedge \forall v_2 \; (node(v_2) \wedge v_2 \not\approx v_1) \rightarrow dom(v_1, v_2)$$

$\mathcal{T}$9. *Exhaustive Ordering*

$$\forall v_1 \, \forall v_2 \; nodes(v_1, v_2) \wedge v_1 \not\approx v_2 \rightarrow$$
$$(\neg(prec(v_1, v_2) \vee prec(v_2, v_1)) \rightarrow (dom(v_1, v_2) \vee dom(v_2, v_1))).$$

$\mathcal{T}$10. *Downward Propagation of Precedence Properties ("DownProp")*

i. $\forall v_1 \, \forall v_2 \, \forall v_3 \; nodes(v_1, v_2, v_3) \rightarrow$
$((dom(v_1, v_2) \wedge prec(v_1, v_3)) \rightarrow prec(v_2, v_3))$

ii. $\forall v_1 \, \forall v_2 \, \forall v_3 \; nodes(v_1, v_2, v_3) \rightarrow$
$((prec(v_3, v_1) \wedge dom(v_1, v_2)) \rightarrow prec(v_3, v_2))$

In addition we need one final axiom, if we want to rule out "dense orders". Given the axioms so far, note that the rational numbers between 1 and 0 model them. The number 1 is the root, there is only one branch, all the rational numbers are ordered, letting arithmetic less-than model the *dom* predicate. However, in this structure no node has a mother. That is, between any pair of "nodes" we can always find another. It is certainly not a property

[14] That is, $distinct(v_1, v_2, v_3)$ is equivalent to $v_1 \not\approx v_2 \wedge v_1 \not\approx v_3 \wedge v_2 \not\approx v_3$.

we would want to hold of trees that between any two nodes we can find an infinite number of other nodes. Dense branches are allowed because I have taken remote dominance as primary and defined immediate dominance in terms of it, allowing thereby a situation in which dominance is well defined but as it happens no pair of nodes satisfies the conditions on immediate dominance.

In such a case we have two choices. First, we could add an axiom like $\mathcal{T}$3011, in (30), below. This simply states that all nodes are either the root (i.e., not dominated by anything) or they are immediately dominated by something, i.e., we insist that every node except the root have a mother.

(2.30) *All nodes but the root have mothers.*

$\mathcal{T}$11. $\forall v_1 \; (\exists v_2 \; i\text{-}dom(v_2, v_1)) \vee (\neg \exists v_2 \; dom(v_2, v_1))$

This succeeds in ruling out densely ordered branches in models of $\mathcal{T}$, but there is another question to address: would such an axiom help a parser to derive interesting facts about trees? Note first that dense orders obviously only arise in infinite trees, so axiom $\mathcal{T}$3011 is only necessary if we expect to derive infinite trees (and wish to rule out dense branches in those infinite trees). In practice all our models will be finite, and we may even want this to be so as a matter of principle, but I will leave that question open here. In any case this axiom would be of little use to us since the existentially bound variable which it introduces into the description could never take part in grammatical relations, since it cannot appear outside the scope of its quantifier.

2.3.5 Deriving Properties of Trees

This is a very compact set of axioms, and the reader should take heart if the familiar properties of trees do not immediately emerge from them. The most important properties can be brought out fairly easily, though. For instance, we do not normally allow loops in trees (we could say equivalently: there ought to be no upward pointing branches). This we get by combining the first two axioms, as in (31). Note that here I am attempting to illustrate the axioms in their role as a definition of tree-hood, not as part of a deductive calculus. Therefore I will give somewhat informal proofs, rather than fully formal deductions.[15]

[15] I will attempt to maintain a terminological distinction throughout this work between "proofs", that is, arguments which mathematicians might advance in order to convince each other of some fact, and, on the other hand, "deductions", as they are defined above on page 31.

Tree structures are a species of directed *acyclic* graph—no loops are allowed. This requirement follows straightforwardly from our axioms of Irreflexivity ($\mathcal{T}285$) and Transitivity ($\mathcal{T}286$).

(2.31) *Asymmetry Theorems*

(a) $\forall v_1 \forall v_2 \neg(dom(v_1, v_2) \wedge dom(v_2, v_1))$
(b) $\forall v_1 \forall v_2 \neg(prec(v_1, v_2) \wedge prec(v_2, v_1))$

For a proof, we assume the contrary and derive a contradiction. That is, we assume initially that, for some a, b, $dom(a,b) \wedge dom(b,a)$. From $\Lambda 142$ (Universal Instantiation) and $\mathcal{T}28(6)$i (Transitivity) we have that $(dom(a,b) \wedge dom(b,a)) \to dom(a,a)$, so, by *modus ponens*, we conclude that $dom(a,a)$. But we also have $\neg dom(a,a)$, from the Irreflexivity axiom ($\mathcal{T}28(5)$i) plus UI ($\Lambda 142$). Therefore we have a *reductio ad absurdum*: $\mathcal{T} \cup \{(dom(a,b) \wedge dom(b,a))\}$ is inconsistent, so $\mathcal{T} \vdash \forall v_1 \forall v_2 \neg(dom(v_1, v_2) \wedge dom(v_2, v_1))$. An analogous demonstration works for the (31b) case.

Another property of trees which we should certainly insist on is a prohibition against crossing branches. We can express this prohibition as follows. If v_1 dominates v_2, and v_3 dominates v_4, and, finally, if v_1 precedes v_3, then v_2 must precede v_4.

(2.32) *Non-Tangling Theorem*

$$\forall v_1 \forall v_2 \forall v_3 \forall v_4 \; nodes(v_1, v_2, v_3, v_4) \to (dom(v_1, v_2) \wedge dom(v_3, v_4) \wedge prec(v_1, v_3) \to prec(v_2, v_4)).$$

The theorem is a straightforward extension of the Downward Propagation of Precedence Properties axiom ($\mathcal{T}2910$). From $prec(v_1, v_3)$ we get $prec(v_1, v_4)$, using $\mathcal{T}29(10)$ii, because v_4 inherits the precedence properties of v_3. And so $prec(v_2, v_4)$, because v_2 inherits the precedence properties of v_1.

Another important property of trees is analogous to the Exhaustive Ordering axiom, and I have accordingly named it the Exclusive Ordering theorem. It states that no two distinct nodes of a tree can stand in more than one relation with each other. Taken together, Exhaustive and Exclusive Ordering entail that every node of a tree is related to every other node in exactly one way.[16]

[16] Partee, ter Meulen & Wall [Partee 1990] present a similar axiomatization of trees. However, in their treatment they take Non-Tangling to be an axiom, and they combine Exhaustive and Exclusive Ordering into a single biconditional axiom (which they refer to as Exclusivity). But as we have just seen, if we generalize the Non-Tangling axiom, which is essentially what DownProp does, then we get Non-Tangling and the properly exclusive direction of the Exclusivity axiom as theorems.

(2.33) *Exclusive Ordering Theorem*

$$\forall v_1 \forall v_2 \; nodes(v_1, v_2) \wedge v_1 \not\approx v_2 \to ((prec(v_1, v_2) \vee prec(v_2, v_1)) \to \neg(dom(v_1, v_2) \vee dom(v_2, v_1))).$$

The proof is again by contradiction. Negating the consequent gives us four cases, for some nodes a and b, witnesses to the assumed counterexample:

$(prec(a,b) \wedge dom(a,b))$,
$(prec(a,b) \wedge dom(b,a))$,
$(prec(b,a) \wedge dom(a,b))$, or
$(prec(b,a) \wedge dom(b,a))$.

The demonstration is identical for each case, so I will only sketch the first one. Assume then that $prec(a,b) \wedge dom(a,b)$. Then, by DownProp ($\mathcal{T}29(10)$i), b inherits the precedence properties of a, so we get $prec(b,b)$, a contradiction to the Irreflexivity axiom ($\mathcal{T}28(5)$i). The other cases proceed analogously.

Finally, there is the generally convergent shape of the dominance relation. We want to avoid V-shaped structures. That is, all nodes lying above a certain node must lie on a single branch of the tree. We can express this property as saying that if two nodes both dominate a third, then they must stand in the dominance relation themselves.

(2.34) *Convergence Theorem*

$$\forall v_1 \forall v_2 \forall v_3 \; (node(v_1, v_2, v_3) \wedge distinct(v_1, v_2, v_3)) \to (dom(v_1, v_3) \wedge dom(v_2, v_3)) \to (dom(v_1, v_2) \vee dom(v_2, v_1)).$$

By the Exclusive Ordering theorem (33) and the following tautology of sentential logic,

$$(\phi \to \neg\psi) \leftrightarrow (\psi \to \neg\phi),$$

we have that

(2.35) $$\forall v_1 \forall v_2 (dom(v_1, v_2) \vee dom(v_2, v_1)) \to \neg(prec(v_1, v_2) \vee prec(v_2, v_1)).$$

Using a proof by contradiction again, we assume the opposite, that

$$prec(a,b) \vee prec(b,a),$$

for some a, b and c, dominated by both a and b. By DownProp ($\mathcal{T}2910$) together with $dom(a,c)$ and the above assumption we have that $prec(c,b) \vee$

$prec(b,c)$. Both disjuncts are inconsistent with the original assumption that $dom(b,c)$, so we obtain (2.35) by a *reductio ad absurdum* argument and hence, by Exclusive Ordering again, we have our theorem.

These examples should suffice to show that $\mathcal{T}$ does indeed give us the desired definition of tree structures.

2.4 Logical Descriptions

In this section we turn to examine in more detail logical representations of syntactic structure.

A set of sentences in a first-order language is commonly called a *theory* (thus, for example, Λ is a theory, $\mathcal{T}$ is a theory, and so is $\Lambda \cup \mathcal{T}$). The most crucial property of theories is that they be *consistent*, according to the following definition.

Def'n 2 *A set Σ of first order formulas is* inconsistent *iff*

$$\Sigma \vdash \phi \text{ and } \Sigma \vdash \neg\phi$$

for some formula ϕ of $\mathcal{L}$.

Given a sound and complete deductive calculus we have that a set of formulas is inconsistent if and only if it has no model. In what follows I will say "a set of sentences Δ is *consistent with* another set Γ" just in case

$$\Delta \cup \Gamma \not\vdash (\phi \wedge \neg\phi)$$

for any ϕ, or, equivalently,

$$\models_{\mathcal{U}} \Delta \cup \Gamma$$

for some structure $\mathcal{U}$.

The operation of the parsing model is determined by the combination of several separate theories. We have already seen Λ and $\mathcal{T}$, the background theories which should not vary in any of our investigations. In addition to this we will have a grammar. We will see a number of examples of licensing grammars in upcoming chapters. For the time being we can just think of the grammar as a theory which determines which structures the language pairs with each input. The input itself represents a fourth theory. I assume that this too is a logical representation output from lower level processes (lexical lookup and inflectional analysis, in particular). Whatever representation the parser constructs is constrained by these four theories in the sense that it must be consistent with these theories.

Among all logical theories I will be particularly concerned with sets of *ground atomic formulas* of $\mathcal{L}$.[17] A formula is *ground* just in case it contains no variables. Thus, though our background theories (Λ, $\mathcal{T}$ and the grammar) utilize the full resources of first-order logic, we will restrict the logical representations which the parser constructs to what are in effect propositional theories. Note that this restriction carries over to the representation of the input. I will refer to this class of representations as *logical descriptions*.

Def'n 3 *A* logical description *("description" for short) is a set of ground atomic formulas of $\mathcal{L}$.*

Because we will not be attempting to make claims here about the proper content of natural language grammars, preferring to take theoretical linguists at their word, it is an important test of the correctness of this parsing model that it produce the analysis which the grammar, as a linguist would interpret it, says it should produce. The linguist's grammar assumes representations which are (or are very like) tree structures, so the correctness of the parser's output depends on its interpretation in terms of these structures.

There are several reasons to attempt parsing with logical representations. Chief among them is that it gives us a way to interpret representations which are *incomplete*. This is particularly useful, as we will see, in constructing formal models of language breakdown, and, at least hypothetically, in dealing with other situations (an incompletely learned grammar, a noisy environment) in which the actions of the parser are underdetermined. It may also be a useful feature in systems which use syntactic information before the parse is complete. All of these ideas require that we be able to interpret (at some level) descriptions which are partial.

The interpretations I am interested in here are interpretations as we have defined them so far, that is, in terms of tree structures. So the problem to be addressed here is just: what tree structures does a (partial) logical description *mean*.

2.4.1 Canonical Models

Suppose that a parser has constructed a logical description. Let us refer to that set as Δ. Now let us consider the class of structures which are models for Δ (call it "Mod Δ"). Mod Δ can be quite large. For one thing, we have placed no restrictions on what kind of objects we want to have in the domain of an LCO structure. These objects could themselves be arbitrarily

[17] Some other names for the same thing are sets of "atomic sentences" or of "positive (ground) literals".

complex sets, for instance. Clearly, even in a theory which could only be satisfied by a structure with a single element (e.g., the theory consisting of the single sentence $\forall v_1\ v_1 \approx n_0$), that single element could be anything. Since we really don't care what the elements of a model are, as long as they have the right relationships to each other, I will adopt a simplifying assumption. I will assume that we are only interested in a particular *canonical* model. Since it makes no difference what elements populate its universe, I will assume that terms of $\mathcal{L}$ denote themselves in this canonical model. This has been tacitly assumed already for elements of **Attribute-Names** and **Value-Names**: $person^{\mathcal{U}}(x)$ should equal the *person* feature of x. If it did not then I should be accused of willfully misleading the reader. I will hereafter extend this assumption to all the terms. Referring to this new special model as $\mathcal{C}$, we thus have the following:

$$n_1^{\mathcal{C}} = n_1$$

and so on. In general:

$$f(t)^{\mathcal{C}} = f^{\mathcal{C}}(t^{\mathcal{C}}) = f(t).$$

What this means is that t and $f(t)$ are elements of $|\mathcal{C}|$, and f denotes that function which takes t to $f(t)$, that is $\langle t, f(t)\rangle \in f^{\mathcal{C}}(= f)$. We also assume that the predicates receive their obvious interpretation, so that, for instance,

$$\models_{\mathcal{C}} dom(n_1, n_2) \text{ iff } \langle n_1, n_2\rangle \in \mathbf{D}.$$

A problem arises in this definition when we consider statements like

(2.36) $$tense(n_1) \approx past.$$

If we take $tense(n_1)^{\mathcal{C}}$ to equal $tense(n_1)$, as the preceding definition says it does, and $past^{\mathcal{C}}$ to equal *past*, then $\mathcal{C}$ does not satisfy (2.36). The two terms on either side of the $\approx$ sign have to denote identical members of $|\mathcal{C}|$ (recall the definition in (1212e) on p. 28), but the element $tense(n_1)$ is not the same element as the element *past*.

This problem arises frequently in constructing models for first order theories with equality (for example in proofs of the completeness of first order logic). The solution I will adopt involves a two step process. Suppose that we have a theory Δ, and we wish to define for it a canonical model. First we replace $\approx$ everywhere in Δ by $=$ to derive a new theory $\Delta_=$. We can always do this (preserving satisfiability as long as Δ is consistent to start with), since $\approx$ is a more restricted relation than $=$. Recall that $t_1 \approx t_2$ just in case $t_1 = t_2$ and $t_1 \neq \bot$. If Δ is consistent, then t_1 cannot equal $\bot$, and so, of course, neither can t_2, since it is identical to t_1. The only element where

$\approx$ differs from $=$ is $\bot$, so as long as Δ contains no instances of $t \approx \bot$ the substitution should cause no problems.

Then we define a temporary initial structure as above, treating equality just like any other 2-place predicate. As noted already this structure does not verify either the source theory or $\Delta_=$, so we need an extra step.

In this second step, we replace $=$ everywhere by another, new 2-place predicate symbol (say E), giving yet another version of the theory, Δ_E. It probably seems laborious to replace $\approx$ with $=$, and then replace $=$ with E. However, we will see below that we need E to be a congruence relation, which $=$ is, but $\approx$ is not, since it is not always a reflexive relation. (In particular it is not the case that $\bot \approx \bot$.) Note that, instead of showing that $\approx$ is the same as $=$ as long as the theory is consistent I could have shown that $\approx$ is reflexive as long as the theory is consistent, allowing us to go directly to Δ_E. The crucial point is that this construction only works if $f(t) \approx \bot$ is not a theorem: it has no model, but $f(t) = \bot$ does. Substituting for $\approx$ any congruence relation (either $=$ or E) in such a situation would make an unsatisfiable theory satisfiable, an unsound result.

The interpretation for E, $E^{\mathcal{C}}$, is constructed in the obvious way: if $t_1 \mathrm{E} t_2$ is in Δ_E, then $\langle t_1, t_2\rangle \in E^{\mathcal{C}}$. The canonical structure for Δ_E now *is* a model for the theory Δ_E. The trick is that being in E is not at all the same thing as being identical members of $|\mathcal{C}|$. E is just an arbitrary relation. We have in effect relaxed the much stricter requirements which hold of equality; no comparable conditions hold of membership in $E^{\mathcal{C}}$. The only condition we have placed on E has been placed indirectly, in that the original theory Δ must have been consistent in the first place, with the result that E inherits the properties of a consistent $\approx$ relation.

Next, because E is a congruence relation (since $=$ is), we can form a "quotient universe modulo E" $|\mathcal{C}|/E$. That is, if $t_1 \mathrm{E} t_2$, then we put both t_1 and t_2 in an equivalence class, and make that equivalence class an element of $|\mathcal{C}|/E$. We write $[x]$ for the equivalence class including x. Obviously, if x is equivalent to y, then we could just as easily refer to the class as $[y]$. I will also write $\mathcal{C}/E$ for the structure whose universe is $|\mathcal{C}|/E$. Hence $|\mathcal{C}/E| = |\mathcal{C}|/E$.

Given a statement like (2.36), repeated here as (2.37),

(2.37) $$tense(n_1) \approx past.$$

we would have that both $tense(n_1)$ and *past* were elements of the equivalence class $[past]$ (equivalently $[tense(n_1)]$). That is, from $tense(n_1) \approx past$, in Δ, we have derived $tense(n_1) = past$ in $\Delta_=$, and thence $tense(n_1)\mathrm{E}past$ in Δ_E. From the canonical model for Δ_E we have constructed the quotient model by placing these two terms in the same equivalence class. We are now almost in a position to make the claim that $\mathcal{C}/E$ is actually the structure we want as the canonical model for the original theory Δ.

Let us call h the natural map from $|\mathcal{C}|$ to $|\mathcal{C}|/E$, which maps every element t of $|\mathcal{C}|$ to $[t]$. Note that h is a homomorphism. Accordingly, by a theorem of first order logic (the Homomorphism Theorem, see [Enderton 1972, p. 91]), if $\mathcal{C}$ is a model for Δ_E with assignment function s, then $\mathcal{C}/E$ is a model for Δ with assignment function $h \circ s$, the composition of s with h.

Now that we know that $\mathcal{C}/E$ is a model for Δ, we should try to find some more direct way of constructing it from Δ, without the necessity of constructing the intermediate theories $\Delta_=$ and Δ_E. An example of such a construction is given below in (38).

(2.38) The Canonical Structure $\mathcal{C}/E$ for Δ

(a) If $node(n) \in \Delta$ then $[n] \in |\mathcal{C}/E|$.

(b) If $(f(t) \approx x) \in \Delta$ then

  i. $f^{\mathcal{C}/E} = f$, that is, $\langle [t], [f(t)] \rangle \in f^{\mathcal{C}/E}$

  ii. $x \in [f(t)]$).

(c) If $dom(x, y) \in \Delta$ then $\langle [x], [y] \rangle \in \mathbf{D}$.

(d) If $prec(x, y) \in \Delta$ then $\langle [x], [y] \rangle \in \mathbf{P}$.

(e) If $i\text{-}dom(x, y) \in \Delta$ then $\langle [x], [y] \rangle \in \mathbf{D}$.

(f) If $i\text{-}prec(x, y) \in \Delta$ then $\langle [x], [y] \rangle \in \mathbf{P}$.

Note that the conversion of the immediate structural relations is considerably simplified by the assumption that Δ is consistent. A quick comparison of the satisfaction conditions on immediate relations (1212f) with the axioms governing the two *imm* predicates ($\mathcal{T}$287) should convince the reader that the bulk of the work of determining satisfaction is done by these axioms, together with the definition of consistency.

We will be particularly interested in what follows in the *least* structure such that (38) holds. In what follows I will often just write "minimal model" for "minimal canonical model". I will from now on just write $\mathcal{C}$ for the (minimal) canonical model $\mathcal{C}/E$. The minimal model for a theory Δ is unique, since the construction of the canonical model according to the strictures of (38) is completely determined by the particular theory Δ. We can think of this model either as the intersection of all the models which satisfy the conditions in (38) or as the model constructed by scanning the elements of Δ and adding the corresponding piece to the model. Having such a construction means that we can take theories in $\mathcal{L}$ as standing for certain structures, namely their least canonical models. In the next section we will consider

minimal models which are in addition *trees*. This will allow us to have a parser construct *theories of trees*, rather than trees themselves, without losing track of the linguistic significance of such a parser's results.

Note that, as mentioned above, this construction is very similar to the last step of a typical completeness proof for first-order logic with equality. This particular version owes a great deal to the presentation of the completeness proof in [Enderton 1972]. It is also very similar to a Herbrand model, which plays a prominent role in the theory of Logic Programming (as well as in some completeness proofs for first order logic, see, e.g., [Fitting 1990]).

2.4.2 Canonical Models and Trees

Nothing guarantees that a canonical model (in particular the *least* canonical model) $\mathcal{C}$ will be a tree, i.e. a structure satisfying $\mathcal{T}$. It is possible to have a description Δ which is consistent with $\mathcal{T}$ but whose canonical model is not a tree. This could happen, to take a simple example, if there was no node in Δ which could count as a root node. No contradiction of Rootedness ($\mathcal{T}$298) is necessarily entailed by this situation. Given such a theory $\Delta \cup \mathcal{T}$, a structure $\mathcal{U}$ could be a model for that theory as long as an arbitrary node could be included in $|\mathcal{U}|$ which fit the bill. However, the conversion to the canonical model $\mathcal{C}$ does not add this arbitrary node, and so there is not necessarily any node in $|\mathcal{C}|$ which would satisfy axiom $\mathcal{T}$298.

The set of logical descriptions whose least canonical models are (a) trees and (b) unique is thus a proper subset of the set of logical descriptions. I will refer to members of this subset as *complete*, and to descriptions which are not complete as *partial*. Recall that the minimal canonical model for a description is unique (it is a function of the description), so if it is a tree, it is a unique tree. A description whose minimal canonical model is a tree thus serves to pick out a single tree which we take to be the structure which the grammar pairs with the input.

It should be clear that, as long as partial descriptions are consistent with the theory of trees, there should be a way of extending them to complete descriptions. Towards this goal, assume that we have a (possibly partial) description Δ. We define a series of descriptions $\Gamma_0, \Gamma_1, \ldots$, where Γ_0 is the description we want to extend (Δ in this case) and $\bigcup_i \Gamma_i$ is a complete description.

In the first stage of the extension we want to identify a pair of distinct nodes mentioned in Δ for which no explicit ordering statements exist in the current Γ_i, that is

(2.39) $$dom(x, y) \quad \notin \quad \Gamma_i,$$

$$
\begin{aligned}
dom(y,x) &\notin \Gamma_i,\\
prec(x,y) &\notin \Gamma_i, \text{ and}\\
prec(y,x) &\notin \Gamma_i.
\end{aligned}
$$

(Note that since the immediate relations entail the partial orders, we have to consider $i\text{-}dom(x,y) \in \Gamma$ as equivalent to $dom(x,y) \in \Gamma$. Note also that the restriction to nodes mentioned in Δ means that if Δ is finite then all its extensions are as well.) We then construct a sort of tree out of Δ in the following way. All the points of this tree are labeled with theories. (In order to avoid confusion with *syntactic* trees I will refer to the nodes of this meta-theoretical construction as *points* from now on.) The root is Δ, and each point has four daughters, each one of which represents the result of adding one of the ordering statements in (2.39), for some pair of nodes which are still unordered in the theory which labels the parent point. One layer of such a tree is shown in Figure 2.7, where Δ, the label of the mother point, has in it nodes x and y not yet ordered with respect to each other (and where $\Delta; \phi$ is the result of adjoining ϕ to Δ as a new element).

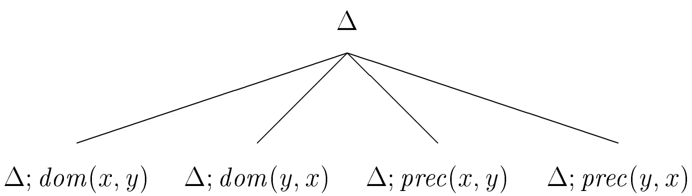

Figure 2.7: Constructing an Extension Tree for Δ.

Let us call a branch *closed* if it contains a point labeled with an inconsistent theory. We say the tree is *closed at* Γ if the description at point Γ is inconsistent with the background theories. For point Γ the tree is closed at Γ just in case it is closed at all of its daughters. The tree represents all the ways of filling out the original description Δ, and we close off and prune away all the ways of doing this which we discover lead to an inconsistent theory.

Because of the way we construct such extension trees, the chain union of all the nodes on an open branch represents an extension of Δ which is *maximal* in the following sense: all the constituent-nodes in Δ are ordered with respect to each other in $\bigcup_i \Delta_i$, and this extension is consistent. This latter follows because every point on an open branch is consistent (otherwise the branch would be closed), and each is a subset of its descendants. Thus, since each Δ_i is consistent, by the definition of an open branch, then the

chain union of these sets is consistent even if the branch is infinite. I will refer to the chain union of the points on an open branch with root Δ as a *maximal extension* of Δ.[18]

I will now argue that the canonical model for each maximal extension of Δ is either a tree, or very nearly so. If it is not a tree, it nonetheless determines a unique tree, which can be constructed from it in a single step.

For every pair of nodes appearing in Δ there is exactly one ordering statement for that pair of nodes on any given branch of the extension-tree. The ordering in the maximal extension is exhaustive by the very method of constructing the tree. It is exclusive by the consistency condition on open branches. Therefore the canonical model for each maximal extension will satisfy all of the tree axioms (with the possible exception of Rootedness ($\mathcal{T}298$)), as I will now attempt to demonstrate.

For the canonical model of a maximal extension to satisfy Reflexivity ($\mathcal{T}285$), it must be the case that the maximal extension does not contain any statement of the form $\pi(x,x)$, for π either *dom* or *prec*. If the source theory contained such a statement, it would be inconsistent, and the tree would be closed at its root. By the rules of its construction no reflexive statement will ever be added to the extension-tree (recall that we only add ordering statements for unordered *distinct* nodes). Therefore the canonical model for a maximal extension will contain no reflexive ordering statements. Note that in any case if the theory came to contain a reflexive statement, it would become inconsistent and the branch is by definition closed at that point.

For the canonical model of a maximal extension to satisfy Transitivity ($\mathcal{T}286$) and Downward Propagation of Precedence Properties ($\mathcal{T}2910$), it must be the case that maximal extensions are deductively closed under these axioms. To see that they are, note that deductions from these axioms yield ordering statements on pairs of nodes from (conjunctions of) ordering statements. Both axioms have roughly the same form, so in general deductions using them will be of the form:

$$P, Q, (P \wedge Q \rightarrow R) \vdash R, \tag{2.40}$$

where P and Q will be found in the label of some point on the branch (since they are atomic ordering statements) and $P \wedge Q \rightarrow R$ can be instantiated as either Transitivity or DownProp. R is an ordering statement, that is $\pi(x,y)$, for π either *dom* or *prec*. By the exhaustiveness of the construction

[18] Note that, if Δ is finite, then all of its maximal extensions will be finite, so taking the chain union is not strictly necessary in such cases. The bottom-most point of an extension-branch will do fine in these finite cases.

some ordering statement on x and y is bound to appear somewhere along the relevant branch as long as x and y are distinct nodes. If that ordering statement is R, then that consequence of Transitivity or DownProp is in the maximal extension. If it is not R, then the branch is closed at that point—by the exclusiveness theorem (33) x and y can be ordered in at most one way. Therefore, any ordering of x and y which is not a consequence of Transitivity and/or DownProp will yield an inconsistent theory, closing the branch. If x and y are not distinct, then of course the theory containing P and Q is inconsistent, and the branch is closed at that point.

Obviously the canonical model of a maximal extension will satisfy the Exhaustive Ordering axiom ($\mathcal{T}$299) by the very exhaustiveness of the construction of an open branch.

To see that this canonical model might still not satisfy the Rootedness axiom ($\mathcal{T}$298), it is helpful to construct the following set. For any open branch B with points $B_1, B_2, \ldots$ let $M = \{x : \text{for no } y \text{ is } dom(y,x) \in \bigcup_i B_i\}$. M is the set of dominance-maximal nodes in $\bigcup_i B_i$. If M is a singleton set, then the canonical model for that branch is rooted, the root being that single member. Otherwise it is not rooted. Furthermore, by the exhaustiveness of the construction all the elements of M must be ordered with respect to each other, so they must be precedence ordered, or they would not be in M. Therefore none of them could serve as a root without contradicting the exclusive ordering theorem (33). In such a case, the least canonical model *which is in addition a tree* will just contain one more node, an element not otherwise occurring in the maximal extension, (let us arbitrarily insist that it be n_i, for the least i such that n_i is not already a node in the maximal extension). In addition we add dominance statements of the form $dom(n_i, x)$, for all nodes x in $\bigcup_i B_i$.

Clearly there is only one way to form this rooted extension, so as a result $\bigcup_i B_i$ has a unique least canonical tree-model. Equally clearly, if $\bigcup_i B_i$ was rooted to begin with, then its least canonical model is (a) a tree and (b) unique. Thus every maximal extension of a consistent theory will serve to pick out a unique tree structure. Thus for any description Δ we have a coherent interpretation which we can use to evaluate a parsing model. If Δ is a complete description, then we interpret it as standing for its least canonical model. If Δ is a partial description, then we interpret it as standing for the set of canonical models picked out by its maximal extensions. Since these models are uniquely determined, we can in practice leave them out of account and consider the meaning of a partial description to be its maximal extensions themselves.

Chapter 3

Licensing Theory

In the previous chapter we examined two theories, Λ and $\mathcal{T}$, together with their models. Suppose that, in addition to these two background theories we have (a) a grammar, G, and (b) a description of some input, I. Both G and I are taken to be theories in $\mathcal{L}$. Let us further restrict I to be a logical description, containing assertions about the lexical items and inflectional elements which lower-level, 'upstream' processors have detected in an input utterance. We take G to be an as yet unspecified theory in $\mathcal{L}$ which will tell us, oversimplifying somewhat, which trees go with which sentences. The parsing problem can then be seen as the problem of finding what trees satisfy the combined theory

$$I \cup G \cup \mathcal{T} \cup \Lambda.$$

If the input sentence has a single reading, then the solution to this problem ought to be unique. If the sentence has multiple readings, then we are clearly interested in discovering all of them, and of course no others. Furthermore, the solution should contain I as a subset in the same way that the terminal string becomes absorbed into the syntactic structure in classical, formal language theoretic treatments.

The problem is thus in many ways similar to the problem of assigning significance to partial descriptions. It is a problem of extending the input in some way so as to yield what are in effect canonical models, but now in addition to satisfying the theory of trees they must also satisfy the conditions of the grammar, a far more complex theory.

In this chapter I will examine the use of the grammar in addressing the parsing problem, and build on the work of the last chapter in such a way as to approach a parsing model for licensing grammars.

3.1 Licensing Grammars and Grammaticality

One characteristic feature of the Principles-and-Parameters framework is the prominence among its principles of *licensing conditions*. An important goal in developing the description language $\mathcal{L}$ has been to provide a platform for investigating the logic of licensing conditions, and of grammars of licensing conditions. It has been a guiding principle of this research that the key to understanding principle-based parsing is to understand the logic of principle-based theories of grammar—in this way we can hope to isolate properties of principle-based grammars which lead to more efficient parsers and better models of human processing, as well as those which make such progress challenging.

The axioms of $\mathcal{T}$ told us what kind of structures we were interested in describing. We now turn to another body of axioms, the grammar, which characterizes which of those structures are well-formed according to a particular natural language. We will now want to say not only that some structure satisfies our notions of tree-hood, but that it satisfies in addition our notions of well-formedness, or grammaticality. Because the grammar is of so much greater interest to us than the more basic notion of generic syntactic structure we will be correspondingly more concerned with the form of the grammar, the forms which its axioms take, and the way they are used than we were with the axioms of $\mathcal{T}$.

In the principles and parameters framework, a syntactic structure is held to be well-formed if every element in that structure is *licensed*.[1] An element is licensed if it satisfies a *licensing condition*. Licensing conditions typically apply only to elements of a particular sort. So, for instance, there is an antecedent-government condition which applies to traces, and a Case-marking condition which licenses noun phrases. Sometimes more than one condition applies to a particular element. Both of the above conditions apply to traces of Wh-moved NPs, for example. Thus an adequate definition of well-formedness must require that an element be licensed by every condition which applies to it.

Note, however, that such a grammar would be vacuously satisfied by an element to which no licensing condition applied: all applicable conditions (i.e. none) would be satisfied. Thus we would admit elements of which the grammar had no knowledge. To avoid this we add that, in addition to satisfying all applicable conditions, every element must satisfy at least one condition.

[1] Cf., e.g., [Chomsky 1986a, Speas 1991].

(3.1) *Well-Formedness in a Licensing Grammar*

Every element of a well-formed structure must be licensed

(a) by every condition which applies to elements of that type, and

(b) by at least one condition.

Note that this definition leaves open not only what licensing is, but also what is meant by "element".

3.1.1 The logical structure of licensing conditions

Licensing conditions have a simple logical structure which suggests a formalized approach to principle-based grammars and parsers. First, every licensing condition contains as a sub-expression a description of the class of elements to which that condition applies. Typically this refers to features of that class, e.g. that its category is NP, or that it is a syntactic operator, or a syntactic affix, but there is no principled reason why it could not refer to a particular structural environment, offering a purely configurational definition of the elements to be licensed.[2]

The rest of the licensing condition is a description of a configuration within which elements of the relevant type are licensed. This usually requires that the element being licensed stand in some particular structural relation with one or more other elements, e.g. the element being licensed must be c-commanded by some other element, within or outside of a domain defined by yet another element, etc.

These two sub-expressions, which I will refer to as the *focus* and the *environment*, define a licensing condition. Logically we can think of the condition as stating that an element E of a tree T is licensed by a condition C just in case E is a member of the class defined by C's focus, and C's environment is satisfied by T. As an example, we might have a condition which states that every lexical head has a maximal projection. The focus of such a condition would specify the features which identify lexical heads, perhaps by stating that the bar level of a node is zero, or that a lexeme feature is defined for that node. The environment for such a condition would state that another node, having the features which distinguish maximal projections, must dominate the first node, and that they must belong to the same projection. A simple

[2] An example might have that all elements of type X which in addition are dominated by elements of type Y must be licensed by in addition sharing some feature with them.

way to do this is illustrated below.[3]

$$\begin{array}{rcl} \text{focus} & = & \{node(X), lxm(X) \approx Y\} \\ \text{environment} & = & \{node(Z), proj(Z) \approx proj(X), Z \approx max(proj(X))\} \end{array}$$

A structure $\mathcal{U}$ will satisfy the focus with an assignment function s if and only if s assigns to X an element of $C_{\mathcal{U}}$ for which the function denoted by lxm is defined. (If lxm is undefined at $\overline{s}(X)$ then $lxm(X) = \perp$, in which case it cannot $\approx$ anything.) The structure $\mathcal{U}$ will in addition satisfy the environment if there is another element of $C_{\mathcal{U}}$, assigned to Z, such that it is in the same projection as X, and in addition it is the maximal element of that projection. (This simple example does not provide appropriate conditions to tell us what we are allowed to conclude about projections or about their maximal elements—the reader is asked to assume that appropriate conditions exist.)

Somewhat informally, I will often write the focus of a licensing condition as a schematic variable φ, and the environment as a schematic variable ε. Then, referring to the element which is the subject of the condition as X, and referring to all the variables other than X which occur free in the focus or environment as $Y_1, \ldots, Y_n$, where these variables are important I will write $\varphi[X, Y_1, \ldots, Y_n]$, for example, for the focus, an expression of $\mathcal{L}$ with variables $X, Y_1, \ldots, Y_n$ occurring free in it, and similarly for the environment. Accordingly, we have as a more formal version of condition (11a), the following.

(3.2) $$(\forall X) \bigwedge_i (\exists Y_{i_1}, \ldots, Y_{i_n})\, \varphi_i[X, Y_{i_1}, \ldots, Y_{i_n}] \rightarrow \varepsilon_i[X, Y_{i_1}, \ldots, Y_{i_n}].$$

We also have the corresponding disjunction, which enforces condition (11b).

(3.3) $$(\forall X) \bigvee_i (\exists Y_{i_1}, \ldots, Y_{i_n})\, \varphi_i[X, Y_{i_1}, \ldots, Y_{i_n}] \wedge \varepsilon_i[X, Y_{i_1}, \ldots, Y_{i_n}].$$

Note that we invariably require grammars to be finite, so these formulas represent strictly finite conjunctions and disjunctions. We think of the fully well-formed syntactic structures of a language as those structures which, in addition to satisfying $\mathcal{T}$ and Λ, satisfy these conditions as well.

It is of course somewhat inelegant to express every licensing condition in the grammar twice (once as an implication in (3.2), and once in the disjunction in (3.3)). It is also somewhat superfluous to express any of the logical

[3] I will by and large assume that the environments of licensing conditions are logical descriptions, that is, sets of atomic sentences. Such a set is logically equivalent to the conjunction of its members, so in effect the language of licensing conditions will for the most part be restricted to conjunction. I will, however, introduce disjunctive licensing environments below (see pp. 70f.).

constants in these conditions, since their logic is fixed, at least at this level of generality. Therefore I will pursue the following strategy. Though we may think of the grammar as a logical theory consisting of (3.2) together with (3.3), I will in general treat it as a set of pairs $\langle \varphi[X], \varepsilon[X] \rangle$, and say that an element E of a model T is licensed by a licensing condition $C = \langle \varphi[X], \varepsilon[X] \rangle$ just in case, when X denotes E, then T is a model for φ and T is a model for ε as well. I define a *licensing grammar* formally as a set of such pairs. In what follows I will sometimes employ the alternative notation $X\hat{}\varphi$ in place of $\varphi[X]$. The symbol '$\hat{}$' is often used in logic programming for lambda extraction (cf. [Pereira 1987]).

3.2 Towards a Parsing Model

In this section we will consider in more detail the parsing problem for licensing grammars. I will suggest several approaches to this problem, and we will see that they all converge.

First, we can consider the problem from a purely logical standpoint as the problem of finding (minimal) models for the collection of constraints in (3.4).

(3.4) $$\models_{\mathcal{S}} I \cup G \cup \mathcal{T} \cup \Lambda.$$

This is essentially the characterization of the parsing problem which I offered at the beginning of this chapter. From this standpoint we might be able to design a parser which worked in a manner similar to the construction of maximal extensions in Ch. 2, §2.4.2. That is, it might extend the input to a maximal extension, but this time using the grammar as well as the tree theory to decide which statements to add. There are some problems with this approach, however. First the grammar is far more complex than the tree theory $\mathcal{T}$. Second, the problem differs from the problem of finding a maximal extension in a crucial way, namely the parser must from time to time introduce new, non-terminal nodes, and functions of non-terminals. The problem is not merely one of deciding which ordering statements to add among a fixed set of nodes, but of deciding when to add new nodes, as well.

I will develop here a relation between logical descriptions which I will write as in (3.5), below.

(3.5) $$\Delta \Rightarrow_c \Gamma$$

This relation holds between two logical descriptions just in case they meet the following conditions.

(3.6) We have that $\Delta \Rightarrow_c \Gamma$ if, for some licensing condition $c = (\varphi_c, \varepsilon_c)$, and some substitution θ of terms in Δ or Γ for variables in φ_c or ε_c (respectively), we have the following:

(a) $\varphi_c\theta \subseteq \Delta$.

(b) $\Gamma = \Delta \cup \varepsilon_c\theta$.

(c) Γ is consistent.

In effect we can think of $\Rightarrow_c$ as a way of extending Δ to Γ by taking a term in Δ to which condition c applies, and adding just enough structure to Δ, to yield Γ, so that that element will get licensed. As such $\Rightarrow$ resembles the extension procedure for partial descriptions of the last chapter, and it also resembles node expansion in a rewrite grammar, if we think of rewriting as in a sense a tree-building operation rather than a relation between strings. The crucial point here is that, because of the close relation between logical descriptions and their minimal canonical models, we may hope to solve the parsing problem as expressed in (3.4) by purely *syntactic*, essentially *deductive* means. The $\Rightarrow$ relation is the first step towards a parsing device which can solve the parsing process by extension, by constructing a logical description which has as its minimal model just the tree which solves the parsing problem for that description.

Above I characterized $\Rightarrow$ as a *relation* rather than a function or operation, and this *caveat* may be required. Since we do not know anything about Γ other than that it contains Δ, there may be a number of possible substitutions for the variables in ε. Nothing prevents Γ from having strictly more terms in it than appear in Δ, so even when a substitution is available of some term in Δ for some variable in ε, it is equally possible to use a term from $\Gamma - \Delta$. Note that as long as we have some strategy which tells us deterministically which instantiations for variables in a licensing environment to try next, we can still speak of licensing as an operation, rather than strictly a relation. Accordingly I will generally refer to a 'licensing operation', by which I mean the logical relation defined in (6) together with a selection function which orders the consideration of the various options made available by the licensing relation.

In general I will assume a conservative strategy in finding instances of a licensing environment: terms in Δ are used as much as possible, and only when no consistent substitution is possible will we consider adding new terms to Γ. This is a greedy algorithm which will have the effect of attaching pieces of structure to sites already present in working memory without considering that the correct attachment site may not arise until later in the parse. In

other words, we should expect this strategy to garden path much more frequently than a human would. Clearly future research will have to address this issue.

I will write $\Delta \Rightarrow \Gamma$ whenever the particular licensing condition employed is unimportant. I will write $\Delta \overset{*}{\Rightarrow} \Gamma$ whenever there is a sequence of licensing steps $\Delta \Rightarrow_{c_1} \ldots \Rightarrow_{c_n} \Gamma$, employing some collection of instances of licensing conditions c_i.

The $\Rightarrow$ relation characterizes the licensing operation on logical descriptions, and captures the basic algorithm underlying all licensing parsers: find an unlicensed element and license it, if necessary building the requisite structure to do so. This operational view also gives us a new perspective on well-formedness: a well-formed structure is one in which all the elements are fully licensed, so it is a description Δ such that:

(3.7) $$\forall\Gamma\ \Delta \Rightarrow_c \Gamma \rightarrow \Gamma = \Delta$$

for all choices of c. The condition in 3.7 should be sufficient to guarantee that all elements will be licensed by some condition. To see that this is so, consider the following argument. If we have a theory Δ which is well-formed according to (3.7), then (3.8) holds.

(3.8) $$\Delta \Rightarrow \Gamma \rightarrow \Delta = \Gamma$$

Take an arbitrary element of Δ, and some condition (φ, ε) which applies to it. Then there is a substitution θ such that

(3.9) $$\varphi\theta \subseteq \Delta,$$

and, in addition,

(3.10) $$\Gamma = \varepsilon\theta \cup \Delta.$$

In consequence, we have that

(3.11) $$\Gamma = \varepsilon\theta \cup \Gamma,$$

by the substitution of equals for equals and the assumption in (3.8). As a result, we have that

(3.12) $$\varepsilon\theta \subseteq \Gamma.$$

And since $\Delta \subseteq \Gamma$, we have in addition, from (3.9), that

(3.13) $$\varphi\theta \subseteq \Gamma.$$

Since the choice of element to be licensed (and the condition which licenses it) is arbitrary, we can claim that a theory meeting (3.7) will also meet condition (3.2), the conjunctive part of our earlier definition of well-formedness.

Of course we have to require in addition that no element in Δ remain unlicensed (condition (3.3), above). The licensing operation is insufficiently powerful to guarantee this. If there is some element of Δ to which no licensing condition applies, that element will persist in all the licensing extensions of Δ. However, note that as long as the grammar does not introduce (via a licensing step) any element of a type not addressed in some licensing condition, and no such element appears in the input, then the licensing operation does all the work we require of it. That is, as long as the grammar-writers do their work well, and inputs always come from within the expected representation space, and the parser correctly implements the licensing operation, then we can relax the parsing problem somewhat and ignore condition (3.3).

In the following sections I will attempt to implement a version of the licensing operation, first in first-order logic, and then in the more restricted subset of FOL which can be handled by the Prolog programming language.

3.2.1 The grammar interpreter as a theory about theories

We are here concerned with the development of parsing models which are deductive calculators able to reason automatically about the structural properties of linguistic expressions. Because the parser embodies a deductive calculus, we must build into it this ability to reason logically about theories (and logical descriptions in particular). That is, the parser must embody a *metatheory*, a *theory about theories*, on which we can use standard techniques of automatic theorem proving to derive the correct extensions of the input description. This metatheory also plays the role of a *grammar interpreter*. Since I will adopt the abbreviatory conventions of §3.1.1 in what follows, that is, I will treat the grammar as a set of pairs of expressions in $\mathcal{L}$, it is necessary for the parser to know about those conventions, and to interpret such a grammar correctly. Accordingly, in the rest of this section I will develop a set of logical statements whose subject matter is sets of atomic sentences of $\mathcal{L}$. I will not attempt to give a full formalization of this new higher-level logic. The purpose of giving such a full characterization of $\mathcal{L}$ was to pave the way for the development of the metatheory, the grammar interpreter. We will not be interpreting the interpreter except insofar as we discuss implementing it as a computer program.

Because the parser is itself implemented in a logic programming framework, it is itself expressed in a first order language, a *metalanguage*. I will accordingly adopt several conventions to make it clear when a formal expression is meant to be in $\mathcal{L}$ or in the metalanguage we use to reason (ultimately automatically) about theories of $\mathcal{L}$. I will write predicates of the meta-system in sans serif face (e.g. well-formed). Following the conventions of the Prolog programming language I will write variables of the metalanguage beginning with a capital letter (e.g. *Cond*, *Elt*). Predicates, functions and constants begin with lower case letters.[4]

I present here more formalized definitions of well-formedness, licensing, and associated notions.

A syntactic structure is *well-formed* just in case it is *fully licensed*, that is, just in case every element in it is *licensed*.

(3.14) $$(\forall Desc)\ \mathsf{well\_formed}(Desc) \leftarrow (\forall Elt)\ (\mathsf{element\_of}(Elt, Desc) \rightarrow \mathsf{licensed}(Elt, Desc)).$$

As far as the definition of an *element* goes, it is generally assumed that it is the constituent nodes which are licensed, that is, that constituency is the right level on which to define these concepts.[5] There is no *a priori* reason why licensing conditions should not apply to pairs of nodes, or relations between pairs of nodes, or other structural configurations. However, since there are very many more pairs of nodes than there are nodes in a tree, this restriction is of considerable computational benefit. Accordingly we will want element_of to be true of *Elt* and *Desc* just in case there is a statement in *Desc* of the form $node(Elt)$.

An element of a syntactic structure is *licensed* in that structure just in case it satisfies every condition which applies to it, and at least one condition applies to it.

(3.15) $$\begin{aligned}&(\forall Elt)(\forall Desc)\ \mathsf{licensed}(Elt, Desc) \leftarrow\\ &\quad (\forall Cond)(\forall Focus)(\forall Envir)\\ &\qquad ((\mathsf{licensing\_condition}(Cond, Focus, Envir) \wedge\\ &\qquad\quad \mathsf{applies\_to}(Focus, Elt, Desc))\\ &\qquad \rightarrow \mathsf{satisfies}(Elt, Envir, Desc)) \wedge\\ &\quad (\exists C')(\exists F')(\exists E')\\ &\qquad (\mathsf{licensing\_condition}(C', F', E') \wedge\\ &\qquad \mathsf{applies\_to}(F', Elt, Desc)).\end{aligned}$$

[4] Note that though I am adopting some of the notation used in writing Prolog programs, much of the presentation of the metalogic presented in this section falls strictly outside the subset of first order logic which Prolog implements. Below in §3.2.2 I will be at some pains to fit the definitions of this section into the kind of formulas Prolog can handle.

[5] As a historical aside, as "sub-atomic" syntactic elements have achieved a prominent role in GB syntax, they have tended to be reanalyzed as separate constituents, as for instance Tense, Agreement, Number and other elements which used to be features and are now often considered heads of functional projections.

Here *Cond* ranges over names of licensing conditions (which may be mnemonic, as θ**_criterion**, or simply arbitrary). This will be useful in keeping lists of which conditions have been satisfied by a particular node, for instance.

Note that it is not necessary in the second, existential subcondition that the element satisfy the condition which applies to it, since if it does not then the element will already have failed the first, universal subcondition.

One thing to keep in mind as the systems we are discussing become ever more abstract is that what are *formulas* in the description language $\mathcal{L}$ become *terms* in the metatheory. Hence when I write something like

$$(\forall Cond)(\exists Focus)(\exists Envir)\ \mathsf{licensing\_condition}(Cond, Focus, Envir)\ldots$$

it is not the case that this is a second-order statement with quantification over formulas. To be sure *Focus* and *Envir* are (sets of) formulas of $\mathcal{L}$. However, as far as the grammar interpreter is concerned they are (sets of) terms.

The predicate licensing_condition illustrated in (3.16) is a schema for the axioms which make up the grammar.

(3.16) $\mathsf{licensing\_condition}(Cond, X\hat{\ }Focus, X\hat{\ }Envir).$

$X\hat{\ }Envir$ and $X\hat{\ }Focus$ are required to be logical descriptions. It may be possible to relax this condition on the focus, but in such a case the work involved in proving that a particular condition applies (or not) to a particular element may increase accordingly. (‘*hat*’ is an infix operator, a function symbol which plays the role of λ abstraction, in effect.)

A licensing condition applies to an element if the element is an *instance* of the focus of the condition. By an instance of a focus $X\hat{\ }\varphi$ we mean that if we substitute n_i for X in φ, then the resulting expression *subsumes* the original description. Formally, for terms, a term t_1 subsumes another term t_2 if there is a substitution of terms for variables in t_1 such that applying that substitution to t_1 yields t_2. For example, $f(X, a)$ subsumes $f(c, a)$ because there is a substitution, namely of c for X, which makes them identical. We say that $f(c, a)$ is an *instance* of $f(X, a)$. We write the substitution as $\{X/c\}$, and, given a substitution θ, we write $t_1\theta$ for the result of applying θ to t_1.[6]

[6] Subsumption is a relation which holds among logical terms according to whether one expression is less general than the other. Thus, roughly, if we have a constant a and a variable X, the variable subsumes the constant, since the reference of the constant is more specific. No two distinct constants subsume one another, but identical constants subsume each other. Any two free variables subsume each other. A complex term subsumes another complex term just in case they have the same function symbol and the same number of arguments, and every argument of the first subsumes the corresponding argument of the second. Thus for example $f(X, a)$ subsumes $f(c, a)$, because $f = f$, X subsumes c, and a subsumes a, but it does not subsume $f(c, b)$, $f(c, a, b)$ or $g(c, a)$.

Subsumption is normally defined on single terms, but here we intend to relate *sets* of terms (i.e. sets of predicates of $\mathcal{L}$), so we need to extend the definition to such objects. One set of terms subsumes another if there is a substitution which can be applied to the first set which yields a subset of the second. We think of applying a substitution to a set as applying that substitution to all the elements in the set. Note that this could result in reducing two elements of the uninstantiated set to a single element in the instantiated one.

The same definition, in terms of subsumption and instantiation, holds for the satisfaction of a condition's environment by a logical description: there must be some subset of the description which is an instance of the environment. The definitions of applies_to and satisfies are given in examples (17) and (18).

(3.17) $(\forall Elt)(\forall Desc)(\forall X)(\forall Focus)$
$\quad \mathsf{applies\_to}(X\hat{\ }Focus, Elt, Desc) \leftarrow$
$\quad\quad X = Elt\ \wedge$
$\quad\quad \mathsf{instance\_of}(Focus, Desc).$

(3.18) $(\forall Elt)(\forall Desc)(\forall X)(\forall Envir)$
$\quad \mathsf{satisfies}(Elt, X\hat{\ }Envir, Desc) \leftarrow$
$\quad\quad X = Elt\ \wedge$
$\quad\quad \mathsf{instance\_of}(Envir, Desc).$

Examples (3.14)–(18) provide a metalogical implementation of the basic logic of the definition in (1). The slight increase in complexity, as compared to the implementation in $\mathcal{L}$ offered in (3.2) and (3.3), is mostly due to the fact that this system is used as a grammar interpreter for licensing grammars written in the (*focus*,*environment*)-notation. This system is not yet a logic of the licensing extension operation $\Rightarrow$, however. The definitions of applies_to and satisfies offered above are quite symmetrical, in that they both rely on instance_of to do the real work. This was not the case in the definition of $\Rightarrow$. There there was an important functional distinction between the treatment of the focus and the environment. The focus was in effect used as a test on a description Δ already given in full detail, whereas the environment was used in constructing a new description Γ, not already given except as the combination of Δ and the (appropriately instantiated) environment. The next section will combine these goals, the metatheoretic treatment of well-formedness and licensing, and the isolation of models by licensing extension, and will in addition recast the metalogical theory of this section in a more restricted, and computationally more tractable, subset of first-order logic, namely the Definite Clause Logic of the Prolog programming language.

3.2.2 A Prolog Implementation

In this section I will present an implementation in the syntax of the Prolog programming language of the grammar interpreter sketched in §3.2.1. This implementation will in addition combine the metatheory of the previous section with the licensing operation of §3.2. The resulting system uses a good deal less machinery than the first-order specifications in §3.2.1, and it is correspondingly easier to follow proofs in it. In the ensuing chapters I will sketch some sample licensing parses using this system and demonstrate that it does in fact implement the basic logic of parsing-by-licensing.[7]

Prolog programs consist of sentences in a subset of first-order logic. The following conventions apply. First, all variables are assumed to be universally quantified at the front of the sentence, so the necessary quantifiers are never written. The clauses that make up a program are all of one of two forms. Either they consist of implications from conjunctions of positive atomic formulas to a single positive atomic formula, or else they consist of a single positive atomic formula. The implication is usually written ":-" rather than "→" (or rather, "←"), and conjunction in the body of the clause is indicated with a comma. "Negation" as it is implemented in Prolog is not the negation of classical first-order logic, but rather "negation as failure". That is, it represents the failure to prove a goal rather than the successful proof of its contrary. So that confusion between these two versions of negation does not arise, we write negation as failure as "\+". The Prolog "=" symbol denotes a relation which is true of a pair of terms just in case they *unify*, that is, there is a substitution which, when applied to both terms, makes them identical.

We will make frequent use of Prolog's list structures. The list containing as elements a, b, c is written $[a, b, c]$. Lists can contain lists, so we could have $[a, [b], c]$, the list which contains a, plus the list containing b, plus c. The empty list is written $[\,]$. We refer to the head and tail of a list using the construction $[H|T]$. If we unify $[H|T]$ with $[a, b, c]$, we will succeed with the substitutions $H = a, T = [b, c]$. Note that if we unify $[H|T]$ with the list $[b]$, we still succeed, with $H = b, T = [\,]$. By convention every list is terminated by the empty list. Confusingly enough, this is not true of the empty list itself: $[H|T]$ does not equal $[\,]$ for any instances of H or T.

The definition of well_formed as it is given in §3.2.1 (and repeated below as (3.19)) has a universal quantifier embedded in its antecedent, and so it falls outside of the subset of first order logical syntax which is implemented in

[7] Note that this system has not been implemented as a computer program. The Prolog notation is essentially promissory, and a demonstration of a certain minimal level of tractability.

Prolog. It also contains an implication where Prolog allows only conjunction.

(3.19) $$(\forall Desc)\ \mathsf{well\_formed}(Desc) \leftarrow$$
$$(\forall Elt)\ (\mathsf{element\_of}(Elt, Desc) \rightarrow \mathsf{licensed}(Elt, Desc)).$$

We can tame these unrulinesses fairly straightforwardly, however. As long as the working description is not infinite we can make a list of all its elements. We can then go through this list, attempting to license each element in it. If an element is not fully licensed we place it on a separate list. At the end of the parse we need only compare the list of unlicensed elements to the empty list $[\,]$. If they are equal, then all elements have been licensed and the structure is well-formed.

(3.20) $$\mathsf{well\_formed}(Input, Desc)\ :-$$
$$\mathsf{unlicensed\_elts}(Input, Desc, UnLicList),$$
$$UnLicList = [\,].$$

We could equally well keep track of which elements *were* fully licensed, and then make certain that the list of licensed elements and the list of all the elements were identical. However it is computationally less expensive to check that a list is not empty than to check that two lists contain the same elements. In particular, if the two lists do not contain their elements in exactly the same order, then checking that they are equal can be quite arduous. Therefore I will pursue this admittedly somewhat backwards approach to well-formedness.

I assume here that the input to the interpreter is itself a logical description, constructed by the adjacent, 'upstream' processing module. In parsing, the input is thus a logical description of the input sentence as it might be output from a lexical lookup and inflectional parsing procedure. As a result the input can be taken to be a subset of the final description, or, from another angle, as the initial state of a licensing derivation as in §3.2. (In fact we need to say a little more, to assure ourselves that the input is not merely a substring of the yield of the structure. I will examine these issues in more detail in §4.1.1, below.) Accordingly, *Input* is passed down to unlicensed_elts, the predicate which actually carries out the recursion on the list of elements, as the current state of the working description.

The definition of licensed in (3.15) of §3.2.1 (repeated here as (3.21)) also falls outside of Prolog's natural abilities, but once again we can change things around a little to make it fit.

(3.21) $(\forall Elt)(\forall Desc)$
  licensed$(Elt, Desc) \leftarrow$
  $(\forall Cond)(\exists X)(\exists Focus)(\exists Envir)$
    (licensing_condition$(Cond, X\hat{}Focus, X\hat{}Envir) \wedge$
    applies_to$(X\hat{}Focus, Elt, Desc)) \rightarrow$
      satisfies$(Elt, X\hat{}Envir, Desc) \wedge$
  $(\exists C')(\exists X)(\exists F')(\exists E')$
    licensing_condition$(C', X\hat{}F', X\hat{}E') \wedge$
    applies_to$(X\hat{}F', Elt, Desc))$.

As with well_formed, as long as the domain of the quantifiers is finite, we can implement quantification by list-building and list-comparison.

The definition of unlicensed_elts/3 (our correspondent to licensed of (3.21)) depends, for purely technical reasons, on a new predicate unlicensed_elts/5.[8]

(3.22) unlicensed_elts(*Desc0*, *Desc*, *UnLicList*) :−
  all_elts(*Desc0*, *EltList*),
  unlicensed_elts(*Desc0*, *Desc*, *EltList*, [ ], *UnLicList*).

It is easiest to think of the fourth argument as an input parameter (here initialized to [ ]) and the fifth argument (here *UnLicList*) as the value the predicate actually computes and outputs.

The first clause of unlicensed_elts/5 ((3.23), below) is just the base of the recursion: when *EltList* has been emptied out, then the recursion ends and the list of unlicensed nodes is passed from the input parameter to the output parameter. The same is done with the working description.

(3.23) unlicensed_elts(*Desc0*, *Desc*, *EltList*, *UnLicList0*, *UnLicList*) :−
  EltList = [ ],
  UnLicList = *UnLicList0*,
  Desc = *Desc0*.

[8] It is standard when discussing Prolog predicates to indicate their "arity", that is, the number of arguments they take. Hence "unlicensed_elts/3" and "unlicensed_elts/5". Predicates with the same principle functor but different arities are treated as different predicates, hence the usage. Here I will only mention the arity of a predicate if there is a risk of confusion between two distinct predicates with the same functor.

The recursive clause of unlicensed_elts/5 is given in (3.24)

(3.24) unlicensed_elts(*Desc0*, *Desc*, *EltList*, *UnLicList0*, *UnLicList*) :−
  select_elt(*EltList*, *E1*, *Es*),
  conds_satd(*E1*, *Desc0*, *Desc1*, *SatCondList*,
    UnSatConds, *NewElts*),
  chk_licd(*E1*, *SatCondList*, *UnSatConds*,
    UnLicList0, *UnLicList1*),
  add_new_elts(*NewElts*, *Es*, *NewEltList*),
  unlicensed_elts(*Desc1*, *Desc*, *NewEltList*,
    UnLicList1, *UnLicList*).

In this clause select_elt is true when *E1* is an element of *EltList*, and *Es* is the remainder of *EltList*. The criteria used to select the next element to license may vary, yielding different parsing behaviors. Several such selection functions are examined in some detail in the next chapter. In addition new elements may be added to the element-list in various orders, so that add_new_elts is also a parameter determining the detailed behavior of the interpreter.

Note that, since initially the working description contains only the input description, the predicate all_elts cannot actually return a list of all the elements that will ultimately need to be licensed. In effect the domain of the universal quantifier is only partially specified throughout most of the parse. The predicate unlicensed_elts/5 is recursive on the length of *EltList*. So note that if for some reason the parser is able to keep generating new nodes to add to this list, then unlicensed_elts/5 will run on forever. (We will see an example of this, a so-called "Left Recursive" context-free grammar, in the next chapter.)

The hard work is done by conds_satd which holds if *SatCondList* is the list of conditions such that they apply to *E1* and *E1* satisfies them, and in addition *UnSatConds* is the list of conditions which apply to *E1*, but which *E1* does not satisfy. The predicate chk_licd just decides whether or not *E1* is fully licensed, based on the contents (or lack thereof) of *SatCondList* and *UnSatConds*. If *E1* fails to be fully licensed, then *UnLicList1* will be *UnLicList0* plus *E1*. Finally there is the recursion step which licenses the remaining elements in *Es*, if possible, and adds them to *UnLicList1* if that is not possible.

Before turning to conds_satd, which really does the work of the predicate licensed in example (3.21) of §3.2.1, I provide the definition of chk_licd.

$$\text{(3.25)}\quad \mathsf{chk\_licd}(\mathit{Elt}, \mathit{SatCondList}, \mathit{UnSatConds}, \mathit{UnLicList0}, \mathit{UnLicList})\ :-\\ \mathit{SatCondList} \backslash= [\,],\\ \mathit{UnSatConds} = [\,],\\ \mathit{UnLicList} = \mathit{UnLicList0}.$$

$$\text{(3.26)}\quad \mathsf{chk\_licd}(\mathit{Elt}, \mathit{SatCondList}, \mathit{UnSatConds}, \mathit{UnLicList0}, \mathit{UnLicList})\ :-\\ \mathit{SatCondList} = [\,],\\ \mathit{UnLicList} = [\mathit{Elt}|\mathit{UnLicList0}].$$

$$\text{(3.27)}\quad \mathsf{chk\_licd}(\mathit{Elt}, \mathit{SatCondList}, \mathit{UnSatConds}, \mathit{UnLicList0}, \mathit{UnLicList})\ :-\\ \mathit{UnSatConds} \backslash= [\,],\\ \mathit{UnLicList} = [\mathit{Elt}|\mathit{UnLicList0}].$$

These clauses enforce the quantified sub-formulas in the definition of licensed in §3.2.1 in the following sense. The way *SatCondList* works is fairly obvious: it enforces the existential condition in the sense that *Elt* is licensed by some condition if and only if *SatCondList* is not empty. The *UnSatConds* list works in a slightly roundabout way: an element is licensed by all applicable conditions if it does not fail to satisfy any applicable condition. We keep track of applicable but unsatisfied conditions in *UnSatConds*, so if it is empty we have that no applicable condition went unsatisfied, that is, all applicable conditions were satisfied. Clause 1 of chk_licd (25) handles the successful case. If either *SatCondList* is empty (26) or *UnSatConds* is not empty (27), then the element is added to the list of unlicensed nodes, and any structure including that element will fail to be well-formed.

Next we examine conds_satd, which comes in two versions: conds_satd/6 (3.28), and conds_satd/10 (3.29–3.32). Conds_satd/6 sets up a recursion on the list of licensing conditions, and conds_satd/10 actually carries out the recursion.

$$\text{(3.28)}\quad \mathsf{conds\_satd}(\mathit{Elt}, \mathit{Desc0}, \mathit{Desc},\\ \mathit{SatCondList}, \mathit{UnSatConds}, \mathit{NewElts})\ :-\\ \mathsf{all\_conds}(\mathit{Conds}),\\ \mathit{SatCondList0} = [\,],\\ \mathit{UnSatConds0} = [\,],\\ \mathit{NewElts0} = [\,],\\ \mathsf{conds\_satd}(\mathit{Elt}, \mathit{Conds}, \mathit{Desc0}, \mathit{Desc},\\ \mathit{SatCondList0}, \mathit{SatCondList},\\ \mathit{UnSatConds0}, \mathit{UnSatConds},\\ \mathit{NewElts0}, \mathit{NewElts}).$$

Once again, because the set of licensing conditions is finite, we can explicitly scan the domain of the quantifiers by recursion on lists, in this case

the list of all licensing conditions (actually the list of the *names* of all the licensing conditions). Clause (3.29) handles the base of the recursion, where the lists constructed during the recursion are passed to the output variables.

$$\text{(3.29)}\quad \mathsf{conds\_satd}(\mathit{Elt}, \mathit{Conds}, \mathit{Desc0}, \mathit{Desc}, \mathit{SatL0}, \mathit{SatL},\\ \mathit{UnSat0}, \mathit{UnSat}, \mathit{Ee0}, \mathit{Ee})\ :-\\ \mathit{Conds} = [\,],\\ \mathit{Desc} = \mathit{Desc0},\\ \mathit{SatL} = \mathit{SatL0},\\ \mathit{UnSat} = \mathit{UnSat0},\\ \mathit{Ee} = \mathit{Ee0}.$$

The recursive clauses of satd_conds/10 deal with three cases. In the first case (3.30), licensing of *Elt* by condition *C1* succeeds and *C1* is added to the list of satisfied conditions.

$$\text{(3.30)}\quad \mathsf{conds\_satd}(\mathit{Elt}, \mathit{Conds}, \mathit{Desc0}, \mathit{Desc}, \mathit{SatL0}, \mathit{SatL},\\ \mathit{UnSat0}, \mathit{UnSat}, \mathit{Ee0}, \mathit{Ee})\ :-\\ \mathit{Conds} = [\mathit{C1}|\mathit{Cs}],\\ \mathsf{licensed}(\mathit{Elt}, \mathit{C1}, \mathit{Desc0}, \mathit{Desc1}, \mathit{Ee0}, \mathit{Ee1}),\\ \mathit{SatL1} = [\mathit{C1}|\mathit{SatL0}],\\ \mathsf{conds\_satd}(\mathit{Elt}, \mathit{Cs}, \mathit{Desc1}, \mathit{Desc}, \mathit{SatL1}, \mathit{SatL},\\ \mathit{UnSat0}, \mathit{UnSat}, \mathit{Ee1}, \mathit{Ee}).$$

In the second clause (3.31) the predicate licensing_fails represents the case where condition *C1* applies to *Elt* but *Elt* does not satisfy its environment in *Desc*. In this case *C1* is added to the list of violated conditions.

$$\text{(3.31)}\quad \mathsf{conds\_satd}(\mathit{Elt}, \mathit{Conds}, \mathit{Desc0}, \mathit{Desc}, \mathit{SatL0}, \mathit{SatL},\\ \mathit{UnSat0}, \mathit{UnSat}, \mathit{Ee0}, \mathit{Ee})\ :-\\ \mathit{Conds} = [\mathit{C1}|\mathit{Cs}],\\ \mathsf{licensing\_fails}(\mathit{Elt}, \mathit{Desc}, \mathit{C1}),\\ \mathit{UnSat1} = [\mathit{C1}|\mathit{UnSat0}],\\ \mathsf{conds\_satd}(\mathit{Elt}, \mathit{Cs}, \mathit{Desc0}, \mathit{Desc}, \mathit{SatL0}, \mathit{SatL},\\ \mathit{UnSat1}, \mathit{UnSat}, \mathit{Ee0}, \mathit{Ee}).$$

In the third case (3.32) the condition simply fails to apply. The various data structures are unaffected, and we simply move on to the next condition.

$$\text{(3.32)}\quad \mathsf{conds\_satd}(\mathit{Elt}, \mathit{Conds}, \mathit{Desc0}, \mathit{Desc}, \mathit{SatL0}, \mathit{SatL},\\ \mathit{UnSat0}, \mathit{UnSat}, \mathit{Ee0}, \mathit{Ee})\ :-\\ \mathit{Conds} = [\mathit{C1}|\mathit{Cs}],\\ \mathsf{no\_licensing}(\mathit{Elt}, \mathit{Desc}, \mathit{C1}),\\ \mathsf{conds\_satd}(\mathit{Elt}, \mathit{Cs}, \mathit{Desc0}, \mathit{Desc}, \mathit{SatL0}, \mathit{SatL},\\ \mathit{UnSat0}, \mathit{UnSat}, \mathit{Ee0}, \mathit{Ee}).$$

The Prolog clauses for checking these three cases are given below. They closely reflect the definition of the licensing operation ($\Rightarrow$). The first case, licensed (33), holds of its arguments in case *Cond* applies to *Elt* in *Desc0*, and a new description, *Desc*, can be constructed from *Desc0* together with the environment of condition *Cond*. If the construction involves the introduction of new elements, then these are entered in the list *NewElts*; otherwise *NewElts* comes back empty.

(3.33)
```
licensed(Elt, Cond, Desc0, Desc, NewElts0, NewElts) :-
        licensing_condition(Cond, X^Focus, X^Envir),
        X = Elt,
        applies_to(Focus, Desc0),
        union*(Envir, Desc0, Desc, NewElts0, NewElts),
        \+ inconsistent(Desc).
```

The predicates applies_to and union* are substitutes for instance_of in the first-order presentation of licensing (§3.2.1). As mentioned before, the operation which here is accomplished by union* is a problem. The operation, which is essentially a relative of set union, is complicated by the fact that, given the presence of variables in *Envir*, we don't know exactly which set it stands for. There could be a fairly large number of substitutions to try before we find all the ways in which the environment can be instantiated in *Desc*.

The sheer number of substitutions which a licensing condition's environment allows is a major source of complexity in the operation of this parsing model. It becomes even more problematical since the result of each substitution must then be checked for consistency. The consistency check, which implements the predicate inconsistent, is the most volatile aspect of this model and the one which presents the greatest challenge to future work on this approach to parsing with licensing conditions. I will offer some discussion of inconsistent here, but first I will present the sister-predicates of licensed: licensing_fails and no_licensing.

In licensing_fails the element being checked matches the focus of condition *Cond*, and an extension of *Desc0* is accordingly built to license it. Licensing fails because the resulting extension (*Desc*) is inconsistent with $\mathcal{T}$ and Λ.

(3.34)
```
licensing_fails(Elt, Desc0, Cond) :-
        licensing_condition(Cond, X^Focus, X^Envir),
        X = Elt,
        applies_to(Focus, Desc),
        union*(Envir, Desc0, Desc, [], NewElts),
        inconsistent(Desc).
```


Note that in this predicate, since *Desc* is constructed only to be thrown away after the consistency check, neither it nor the new elements list are passed as arguments of the clause-head.

The predicate no_licensing holds just in case condition *Cond* simply fails to apply to *Elt* at all.

(3.35)
```
no_licensing(Elt, Desc, Cond) :-
        licensing_condition(Cond, X^Focus, X^Envir),
        X = Elt,
        \+ applies_to(Focus, Desc).
```

The logic behind inconsistent is fairly simple: ideally, *Desc*, the result of union* applied to *Envir* and *Desc0*, is combined with the three theory $\mathcal{T}$(and the background logical axioms in Λ), and if the result can prove a contradiction, then the substitution applied by union* must be abandoned and an alternative tried, as long as any remain. In order to ascertain that *Desc* is inconsistent it is thus apparently necessary to take each element p of *Desc* in turn and see if $\neg p$ is a consequence of $Desc \cup \mathcal{T} \cup \Lambda$. Obviously, doing this for every element of *Desc* every time new elements are added to it is computationally prohibitively expensive.

Two goals dominate work on efficient consistency checks. The first is to prevent needless effort. If we do nothing to limit which formulas the consistency check derives a contradiction from, then there is in fact no limit to the work the consistency check must do. Consider what ought to be the best case, when the theory is consistent. Failing to derive a contradiction of any of the assertions in the description, nothing prevents a full first-order theorem prover from deriving consequences of the theory, and attempting to contradict *them*. The consequences of any description will be infinite in number, including many complex tautologies which would seem to the human observer simply inane. Hence the first goal of a working consistency theory must be to limit the class of formulas which are considered for contradiction.

In addition, even restricting our attention to those formulas of $\mathcal{L}$ actually present in a description, some regions of the description will be unaffected by the introduction of new information, and so, if they were consistent before, they will remain so. In other words, we should try to assure that the consistency check examines only those parts of the description which are actually affected by the introduction of new information. Ideally we ought to be able to prove that a limited neighborhood of newly added material is all that can be affected. However, in practice even empirical observations of a running program should be helpful in learning how to restrain the consistency check.

Another source of needless effort lies in dealing with literals which are not in *Delta* to begin with, but which are consequences of it. The same consequence may be derived over and over again as the consistency checker attempt to find various different contradictions. It would certainly be advantageous to keep a record of consequences which have been derived previously. This should shorten the lengths of proofs, and may in addition increase the number of proofs that p is inconsistent, for some p, and hence increase the likelihood of finding such a proof.

The second goal, or class of goals, involves assuring that the consistency check is simple enough to do its work quickly, since it must be called repeatedly during the parse. In particular, it may be that, in a parsing application built around a carefully designed grammar, not all parts of the theory of trees are actually used in determining consistency. To take a very simple example, $\mathcal{T}222$ rules out 'improper' terms in which a feature attribute function is applied to a feature value constant (e.g., $person(3rd)$). However, it may well be the case that no such term could ever arise from using a grammar in which no such terms occurred in the first place. In such a case it would be a waste of resources to check a description for consistency with this axiom. As another example, the reader may recall from the discussion of maximal extensions that the failure of a maximal extension to include a root element has no effect on its ability to pick out a unique canonical model. The analogous conclusion holds for parsing: if there is no node in a description which dominates all other nodes, it does not follow that the description is inconsistent. The addition of a single new node and appropriate dominance statements is sufficient to assure compliance with the Rootedness axiom.

Removing elements of the tree theory in the parsing implementation results in a proof system which is still sound, but no longer complete. This should be fairly obvious: with a subset of the original axioms, all the theorems which could be derived using only that subset remain theorems, but we lose all those theorems which made crucial use of one of the lost axioms. Another way in which the consistency check can be tuned specifically to the parsing application involves replacing axioms with theorems of $\mathcal{T}$. Once again, since the theorems are consequences of the original axioms, then consequences of the theorems must be consequences of the original axioms as well, so such a re-axiomatization, where old theorems become new axioms, is once again sound, but not necessarily complete. As an example, in practice (as we will see in the next chapter) the most common illegal configurations which actually arise in parsing violate either the Exclusive Ordering theorem (33) or the prohibition on crossing branches (32).

Another possible benefit of removing certain axioms involves the decidability of the fragment of first-order logic implemented in the consistency checker. For example, if we attempt to recast the axiom of Exhaustive Ordering in clausal form, as in 36, below, then we note immediately that it is

not Horn—it has more than one positive literal.

(3.36) $$\{\neg nodes(v_1, v_2), v_1 \approx v_2, \\ prec(v_1, v_2), prec(v_2, v_1), dom(v_1, v_2), dom(v_2, v_1)\}$$

Similarly, the axiom of Rootedness is not Horn. It gives us the two clauses, the second of which is non-Horn:

(3.37) $$\{node(c)\}, \{\neg node(v_2), v_2 \approx c, dom(c, v_2)\}$$

In addition it introduces a Skolem constant, c, for the existentially bound variable v_1 of $\mathcal{T}298$. Since we are not entitled to conclude anything about this constant other than what the theory gives us explicitly, we are not allowed to conclude that it equals some other node or has any features, for example. I have argued above that this axiom may not be required in the consistency check. If the same can be argued for Exhaustiveness then we can move the logic of consistency that much closer to the Horn subset of first order logic, which has the advantage of being decidable. Along these lines, note for example that Exhaustiveness is not used in the proofs of any of the theorems of §2.3.5. Its cousin, the Exclusive Ordering theorem, is far more useful in practice, and has the extra advantage that it can be cast in a form that is Horn.[9]

The subject of consistency checking for logical descriptions is worth a (computer science) dissertation all on its own, so I will not attempt to develop the subject more here, except to note one final source of increased efficiency. It is possible (see, for example, [Stabler 1992]) to absorb the axioms of the consistency theory into a specially designed proof rule. Thus, we can describe certain configurations appearing in descriptions which we know we can derive contradictions from. For example, crossing branches, or clashing features or multiply ordered nodes all arise again and again in parses (as we will see in the next chapter). These ultimately yield contradictions of $\mathcal{T}$, but the description of these configurations requires only positive literals. Thus we can hope to design a special purpose proof rule which becomes in effect a pattern-matcher,

[9] Note that that form is Horn, but not definite. Its clausal form consists of a set of clauses roughly of the form:

$$\{\neg dom(X,Y), \quad \neg prec(X,Y)\}, \\ \{\neg dom(X,Y), \quad \neg prec(Y,X)\}.$$

This is indeed Horn: each clause has at most one positive literal (in fact it has none), but it is not Definite, because it does not have exactly one positive literal. Theorem provers for theories which are definite but not Horn exist (see for example the Hornlog system of [Gallier 19??, Raatz 1990]). Note, however, that Prolog is a Definite clause theorem prover, since only the goal clause can be totally negative.

hunting for certain patterns in a description. Since no negative literals are required in these patterns (if they were, we could not match them against logical descriptions!), we have only to consider the atomic consequences of a theory, which, given our theory of trees, will always be finite if the theory is finite. Such a consistency checker will thus always halt.

I will close this section with a simple example which implements some of the ideas discussed above. This implementation of a consistency checker, though it is quite simple, is nonetheless sufficient to catch instances of crossing branches and multiply ordered nodes. The topmost clause, inconsistent, is in a sense an example of a specialized proof rule, which, instead of looking for instances of $p \wedge \neg p$, looks for instances of reflexive ordering statements. This is because it is in fact much easier to derive positive literals from a logical description than to derive negative ones, and, of course, the derivation of a reflexive ordering statement leads immediately to a contradiction by application of an instance of the Irreflexivity axiom.

(3.38) $\mathsf{inconsistent}(T)\ :-$
$\qquad \mathsf{consequence}(dom(X,X),T).$

(3.39) $\mathsf{inconsistent}(T)\ :-$
$\qquad \mathsf{consequence}(prec(X,X),T).$

The predicate consequence is actually limited to finding *atomic* consequences. Since this system is set up to trap descriptions with reflexive consequences, which are atomic, this appears to be sufficient. The predicate works by searching the theory for a desired consequence and, if it is not there, it adds new consequences to the theory until it finds it.[10]

(3.40) $\mathsf{consequence}(P,T)\ :-$
$\qquad \mathsf{member}(P,T).$

(3.41) $\mathsf{consequence}(P,T)\ :-$
$\qquad \backslash+\ \mathsf{member}(P,T),$
$\qquad \mathsf{add\_consequence}(T,T1),$
$\qquad \mathsf{consequence}(P,T1).$

Consequences are added to the theory by applying modus ponens, using either Transitivity or DownProp as axioms. We could equally well think of

[10] Note that this implementation is extremely inefficient since it may add the same consequence many times if that consequence has many proofs. This particular problem is easy enough to solve, as it happens. We can prevent a consequence from being added more once by adding a Prolog cut operator (!) after the call to add_consequence in (41). This vastly improves the performance of the system.

the consistency checker as a Gentzen style natural deduction system, with Transitivity and DownProp as its proof rules.

(3.42) $\mathsf{add\_consequence}(T0,[Cons|T0])\ :-$
$\qquad \mathsf{axiom}(Ante, Cons),$
$\qquad \mathsf{subset}(Ante, T0),$
$\qquad \backslash+\ \mathsf{member}(Cons, T0).$

(3.43) $\mathsf{axiom}([dom(X,Y), dom(Y,Z)], dom(X,Z)).$

(3.44) $\mathsf{axiom}([prec(X,Y), prec(Y,Z)], prec(X,Z)).$

(3.45) $\mathsf{axiom}([dom(X,Y), prec(X,Z)], prec(Y,Z)).$

(3.46) $\mathsf{axiom}([dom(X,Y), prec(Z,X)], prec(Z,Y)).$

One can immediately see ways in which this consistency checker can be improved. For one thing, as written, it builds in the consequences of a theory until it finds a contradiction, or can no longer find new consequences, and then it discards this extended theory. It should be possible, and certainly would be desirable, to retain the consequences of a consistent theory, since many of them will be needed again upon the modification of that theory by further parsing steps. Nonetheless, the fact that such a simple device can spot the most common inconsistent configurations fairly efficiently is grounds for hope that the strategies surveyed above can yield usable consistency checks.

3.2.3 A Note on Disjunctive Licensing Environments

It is pretty much impossible to write an interesting grammar for even the smallest fragment of a natural language without introducing disjunction. To state the ways in which a VP can be expanded in a context-free grammar (as V, V NP, V PP, V NP PP,...), or the ways in which an NP can receive abstract Case in a GB grammar (inherently, structurally under government, structurally via agreement, by binding a trace in a Case-marked position,...) is impossible without being able to enumerate a number of options. This problem cannot be avoided in the presentation of licensing grammars.

It can, however, be minimized. Accordingly I will place the following weakened constraint on the form of licensing conditions: they must consist of a focus which is a logical description and an environment which is either a logical description or a set of logical descriptions, in which latter case the elements of that set are interpreted disjunctively. In effect this means we can write licensing environments using both conjunction and disjunction, but to record them in the grammar we must first convert them to disjunctive normal form. Note that there is no obvious reason not to allow disjunctive foci as well, but they do not appear to come up as often, and can in any case always be treated by writing multiple conditions, one for each focus-disjunct.

Declaratively, a disjunctive environment $\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k\}$ is interpreted as satisfied by a logical description if some subset of the description is an instance of any one of the ε_i. Procedurally, if an element is an instance of the focus in some licensing description, then that description can be expanded by adding any one of the disjuncts in the environment, so long as the result is consistent. In the Prolog implementation this would mean that `union*` would have to be able to recognize the case where *Envir* was disjunctive and, in this case, go into a recursion on the list of ε_i, calling its non-disjunctive version on each of the disjuncts.

Given that we already have potentially large inefficiencies just in searching the space of possible substitutions for the environment, obviously multiplying this by adding in even more disjunction compounds the problem, and we must be very careful in any implementation. One apparently desirable goal would appear to be to interleave these two searches, so that each disjunct is checked using the first substitution before the next substitution is examined.

Chapter 4

"Context-Free" Licensing Grammars

The claims I am pushing in this dissertation are that Licensing Grammars can express a substantial chunk of GB principles, and that Logical Descriptions are a good way to parse with Licensing Grammars. If we allow ourselves to derive looser, partial, remote-dominance based representations from sets of licensing conditions, the argument goes, then we can extract structure more directly from licensing conditions.

Licensing conditions contain structural information, information about the structural configuration in which the licensing relation they specify is established. Our goal is to find ways to extract that information efficiently. The closer that structural information is to being a syntactic tree, the easier it is to extract and use in constructing parse-trees. The conditions or rules that make up a phrase structure grammar, such as a context free grammar, specify configurations which are complete little trees, consisting of a root and its immediate daughters. As we will see in this chapter it is quite easy to extract structure from such conditions, and in fact the actions of a licensing parser, given such a grammar, are almost point for point identical to a more traditional, tree-building parser. Principles and parameters grammars such as those cast within the GB framework consist of conditions which specify syntactic structure much more indirectly, via a system of logical constraints. The goal of the Licensing Grammar formalism as I am developing it in this dissertation is to present a way of extracting structure from licensing conditions such as are used in GB theory. These structural configurations are not necessarily very tree-like. However, they can be captured in a more general representation, logical descriptions, which can be interpreted via the semantics of §2.2 as systems of constraints, and which can also be interpreted, via the notion of canonical models and maximal extensions developed in §2.4, as trees or sets of trees.

If Licensing Grammars and Logical Descriptions are powerful enough to represent the kind of loose, underspecified structural configurations which support the licensing relations specified by a GB grammar, then they should be powerful enough to represent the kinds of tight, exhaustively specified structural fragments which support the *is-a* or *rewrites-as* relations which context free productions specify. In this chapter I will attempt to demonstrate this, using context-free grammars and parsers as a rich source of familiar examples for developing licensing grammars and showing how the interpreter of the previous chapter parses with them.

The relation between CFGs and 'CFLGs' is not difficult to demonstrate, and in fact it appears possible to convert a CFG to an LG mechanically, though the result loses a great deal of the succinctness which is such an appealing property of CFGs, and the conversion looks like it can be fairly arduous to perform in detail. Nonetheless the investigation adds a good deal to our understanding of Licensing Grammars and the properties of the licensing parser. I will go through some sample parses, following the logic laid out in the previous chapter. In ensuing chapters I will extend this investigation to more general licensing grammars with the kinds of conditions we will need to implement significant fragments of GB Theory. If we can make any kind of interesting use of the LG formalism in expressing principles of GB theory, then we may be able to establish a bridge between phrase structure grammars, which have been subjected to intensive and highly successful parsing theoretic investigation, and principles and parameters grammars, for which the parsing problem is extremely challenging, and parsing theoretic investigation is still pretty much in its infancy.

4.1 Licensing from Below and Top-Down Parsing

The first question to address is how to interpret a context-free production as a licensing condition. It seems intuitively obvious that rewrite rules can be considered as a kind of constraint, a kind of well-formedness condition on structures, since they dictate where one kind of node can appear in relation to the positions of other kinds of nodes.[1] I have already noted the intuition

[1] The use of both context-free and context-sensitive rules as node admissibility conditions has been explored in work by McCawley [McCawley 1968], who showed that the two interpretations were equivalent for context-free grammars. Peters and Ritchie [Peters 1973] then showed the surprising result that context-sensitive grammars, interpreted as node admissibility conditions, yielded only the context-free languages. More recent work addressing these issues, and references, can be found in [Jacobson 1990].

that applying the constructive licensing operation to a node is similar to expanding that node in a phrase structure derivation or parse. Thus, given a sample grammar like CFG_0 in (1), we ought to expect to be able to recast it in terms of licensing conditions, licensing a node in the corresponding CFLG where it is expanded in the CFG.

(4.1) CFG_0

(a) S $\longrightarrow$ NP VP
(b) NP $\longrightarrow$ Det N
(c) VP $\longrightarrow$ V NP
(d) Det $\longrightarrow$ *the*
(e) N $\longrightarrow$ *boy* | *girl*
(f) V $\longrightarrow$ *kissed* | *kicked*

As a first approximation to the corresponding CFLG, I will treat productions like (1b) as licensing conditions on nodes of category NP, as in (2). I will in general write licensing conditions in a grammar-rule-like format (as in (2) below), rather than as the unit clauses which the interpreter actually uses. (Recall the definition of licensing_condition on pg. 57.) If, as is often the case, the focus consists of a single formula, I will generally omit the set-braces, even though it represents a singleton logical description. The appearance of the $\Rightarrow$ arrow is intended to recall the way licensing conditions are used in the constructive licensing relation of §3.2.

(4.2) $$\begin{aligned}&\mathit{cat}(X) \approx \mathit{np} \Rightarrow\\ &\{\mathit{i\text{-}dom}(X,Y), \mathit{i\text{-}dom}(X,Z), \mathit{i\text{-}prec}(Y,Z),\\ &\ \mathit{cat}(Y) \approx \mathit{det}, \mathit{cat}(Z) \approx n\}\end{aligned}$$

In a sense this is clearly correct. According to grammar CFG_0 an NP can appear if it is supported by the sequence Det⁀N. We can think of this as licensing the NP 'from below'. However, in a sense, this is only part of the story. Equally clearly an NP can only appear in a structure if it is either part of an S or a VP. That is, the NP is also licensed 'from above'. As it happens, however, the way the licensing mechanism 'expands' the nodes it licenses is sufficiently close to the way that the rewriting mechanism expands nodes into strings of nodes to simulate most of a top-down derivation with just principles 'licensing from below', as in CFLG_0, below.

(4.3) *CFLG*$_0$

(a) $cat(X) \approx s \Rightarrow$
$\{i\text{-}dom(X,Y), i\text{-}dom(X,Z), i\text{-}prec(Y,Z),$
$cat(Y) \approx np, cat(Z) \approx vp\}$

(b) $cat(X) \approx np \Rightarrow$
$\{i\text{-}dom(X,Y), i\text{-}dom(X,Z), i\text{-}prec(Y,Z),$
$cat(Y) \approx det, cat(Z) \approx n\}$

(c) $cat(X) \approx vp \Rightarrow$
$\{i\text{-}dom(X,Y), i\text{-}dom(X,Z), i\text{-}prec(Y,Z),$
$cat(Y) \approx v, cat(Z) \approx np\}$

(d) $cat(X) \approx det \Rightarrow$
$\{i\text{-}dom(X,Y), cat(Y) \approx the\}$

(e) $cat(X) \approx n \Rightarrow$
$\{i\text{-}dom(X,Y), cat(Y) \approx boy\} \vee$
$\{i\text{-}dom(X,Y), cat(Y) \approx girl\}$

(f) $cat(X) \approx v \Rightarrow$
$\{i\text{-}dom(X,Y), cat(Y) \approx kissed\} \vee$
$\{i\text{-}dom(X,Y), cat(Y) \approx kicked\}$

In general, where we have a CF production like (4.4), below,

(4.4) $$A \longrightarrow B_1 \ldots B_k$$

we will have a licensing condition of the form in (4.5).

(4.5) $$\begin{aligned} & cat(X) \approx a \Rightarrow \\ & \{i\text{-}dom(X,Y_1), \ldots, i\text{-}dom(X,Y_k), \\ & i\text{-}prec(Y_1,Y_2), \ldots, i\text{-}prec(Y_{k-1},Y_k), \\ & cat(Y_1) \approx b_1, \ldots, cat(Y_k) \approx b_k\} \end{aligned}$$

Where a category A is expanded as a single daughter B, there should of course be no precedence statements in the corresponding licensing condition.

We see immediately that the licensing grammar formalism is a fairly cumbersome way to express the information contained in the corresponding CFGs. This is not a result of using of a logical language to express the grammar—there are very many quite compact logic-based grammar formalisms for CFGs and their extensions, in fact they abound in the prolog literature.[2] It is a side effect of the explicitness of the formalism, rather. Its

[2] The classic example is the Definite Clause Grammar (DCG) formalism of Pereira and Warren [Pereira 1980]. If this is restricted to context-free resources, then a DCG exactly resembles the CFG formalism.

lack of an interpreter specifically attuned to context-free grammars means that the grammar rules have to be more explicit about how they want themselves interpreted.

In a rewrite grammar, all the dominance statements are taken care of in the single 'rewrites-as' relation, and the precedence statements are taken care of in the left-to-right writing conventions for strings. Explicit reference to nodes or node-variables is not necessary, since different occurrences of the same category (e.g. NP) will never be confused. Each instance of a category is expanded at most once, so in effect individual uses of the rewrite rule are sufficient to identify individual nodes. These are essential properties of context-free grammars, indeed of string rewrite grammars in general. The Licensing Grammar formalism was designed as an aid in investigating grammars which do not have these properties. Suppose we ask the simple question: how many rules does a given node trigger? For a CFG the answer is "at most one" (the terminals do not trigger any rules, otherwise the answer is "exactly one"). For GB theory the answer depends on the type of node, and is in general greater than one. For example, a lexical head node may contribute information about its phrasal projection, the structure within which it assigns thematic roles to its arguments, the structure within which it Case-marks its neighbors, the agreement features of its Specifier, features of its governor (Case-assignment properties in the case of nouns, tense features in the case of verbs). The upshot is that nodes get reused many times, and the structural relations between them must be expressed without prejudice to what future rule-applications will do to them. Hence we are required to be able to represent syntactic structures with remote dominance, and immediate dominance, the rule with CFGs, is somewhat exceptional in a GB-style grammar. I will treat these issues in more detail in the next chapter. Here it is sufficient to point out that there is a reason why CFGs seem so unnatural when recast as CFLGs. From the standpoint of a theory oriented to such long-distance structural relations as government, projection, movement and binding the rules of a CFG *are* somewhat unnatural. Since the Licensing Grammar formalism was designed to capture grammars whose statements lack this restricted sphere of influence, it requires the explicit statement of information such as dominance, precedence and specific reference to nodes. Though this leads to major redundancies in the statement of CFLGs, the presence of such redundancies cannot be presumed of other sorts of licensing conditions, in particular those which appear in GB theories.

Now suppose we want to parse the sentence in (6) using the licensing grammar CFLG$_0$ and the interpreter of Chapter 3.

(4.6) *The boy kissed the girl.*

The input to the parsing model is represented as a logical description, as in example (4.7).

(4.7) $\{cat(n_1) \approx the, cat(n_2) \approx boy, cat(n_3) \approx kissed,$
$cat(n_4) \approx the, cat(n_5) \approx girl,$
$i\text{-}prec(n_1, n_2), i\text{-}prec(n_2, n_3),$
$i\text{-}prec(n_3, n_4), i\text{-}prec(n_4, n_5)\}$

In general, where we have a string of words $w_1 \ldots w_n$, we will have a corresponding description as in (4.8).

(4.8) $\{cat(V_1) \approx w_1, \ldots, cat(V_n) \approx w_n,$
$i\text{-}prec(V_1, V_2), \ldots, i\text{-}prec(V_{n-1}, V_n)\}$

Other information might be included, such as the transitive closure of the *prec* relation. However such information is unnecessary, since we have transitivity as an axiom. (Even in the LG formalism there are some redundancies we can factor out!)

As given, the input description offers nothing to trigger any of the licensing conditions, so there is nothing the parsing model can do.[3] However, since we are mimicking a top-down parse, which knows the category it wants to assign to the input right from the start, it is reasonable to suppose that in addition to a description of the input sentence, our licensing parser starts out with the knowledge that the input string should be an S. Accordingly we add an S-node to the initial description, here n_6.

(4.9) $\{cat(n_1) \approx the, cat(n_2) \approx boy, cat(n_3) \approx kissed,$
$cat(n_4) \approx the, cat(n_5) \approx girl,$
$i\text{-}prec(n_1, n_2), i\text{-}prec(n_2, n_3),$
$i\text{-}prec(n_3, n_4), i\text{-}prec(n_4, n_5),$
$cat(n_6) \approx s\}$

We could add that n_6 dominates all the other nodes currently in the description. However, given the form of the licensing conditions in CFLG_0 this will prove to be unnecessary. There is no other place to start but at n_6, and since all licensing is upwards, all licensee-expansion will be downwards, and n_6 will thus end up as the root without any further effort on our part.

The goal of the parser, as sketched in the previous chapter, is to extract a list of unlicensed elements, and check that it is empty. The first step

[3] This is actually symptomatic of a deeper problem, which I will take up below, since there is no way with this grammar to assure that every node in the description is licensed—the terminals will always be an exception to this condition.

(applying unlicensed_elts/3 in (3.22)) is to collect a list of the elements in the initial description. This gives us a partial description of the domain of the quantifiers which appear in the licensing logic. I will treat this data structure as a stack, for now, with new nodes added at the front. I will refer to this list as *EltList*. It is the list of elements which the parser works through, in particular in the predicate unlicensed_elts. In this chapter I will assume that elements when they are licensed are moved to a list of licensed elements, which I will refer to as *Ls*. This serves no purpose in the parser (though in a sense it is the complement of the list of unlicensed elements which the parser actually builds), but it may be of potential use to the reader in keeping track of what work has been done, and what order nodes have been processed in. Initially, then, we have *EltList* and *Ls* as in (4.10).

(4.10)
$$\begin{aligned} EltList &= [n_6, n_1, n_2, n_3, n_4, n_5] \\ Ls &= [\,] \end{aligned}$$

The predicate select_elt removes the top element of *EltList*, in this case n_6. The predicate conds_satd then attempts to discover which conditions n_6 satisfies, and which conditions apply to it, but it fails to satisfy. (Since the parser does nothing when a condition fails to apply to a node, I will simply ignore those steps.) Our first step is to see which conditions apply. That is, we need to find a licensing condition in CFLG_0 into whose focus we can substitute n_6. (Note that, as discussed above, as long as we are dealing with string-rewrite grammars there should only be one licensing condition which applies to any given node.) Condition (33a) fits the bill. The result of the substitution of n_6 for X in (33a), given in (4.11), fits (exactly) what (little) we know about n_6.

(4.11) $(cat(X) \approx s)\{X/n_6\} = (cat(n_6) \approx s)$

Carrying the substitution through the environment of (33a), we fail to find elements already in the description which fit the requirements of n_6's licensing environment. All elements in the description are already provided with category features which clash with the features that condition (33a) would impose on them. The parser searches the members of the description for atoms matching those in (3a), in particular $cat(Y) \approx np$ and $cat(Z) \approx vp$. No formulas in the description unify with these. (Because prolog's unification algorithm implements syntactic equality for constants, we can use it to implement the consistency condition $\mathcal{T}191$, the Distinctness of Feature Values.) Accordingly union* can only succeed by adding new fresh terms to the description as witnesses of the licensing requirements on n_6. Referring to the environment of (33a) as ε_{3a}, and to the substitution $\{X/n_6, Y/n_7, Z/n_8\}$ as θ_1, we have the following.

(4.12)
$$\begin{aligned}\varepsilon_{3a}\theta_1 = \ & \{\textit{i-dom}(n_6, n_7), \textit{i-dom}(n_6, n_8),\\ & \textit{i-prec}(n_7, n_8),\\ & \textit{cat}(n_7) \approx \textit{np}, \textit{cat}(n_8) \approx \textit{vp}\}\end{aligned}$$

All of the information in (4.12) is new to the description, so all of it is added. It should be clear that the resulting description is consistent. The name of condition (3a) is added to the satisfied conditions list by conds_satd/10 (clause 3.30). (Here I will assume that the relevant example number, e.g. (3a), will do for the condition name.)

No other conditions apply, as we would expect, so the recursion on the list of conditions bottoms out, and the parser returns to unlicensed_elts/5 with (3a) on its satisfied conditions list, and nothing on its unsatisfied conditions list. Accordingly chk_licd will not add n_6 to the unlicensed elements list. We simply remove the now-fully-licensed n_6 from the element-list. The new nodes added in this step are placed at the top of the element-stack by add_new_elts.

(4.13)
$$\begin{aligned}\textit{EltList} &= [n_7, n_8, n_1, n_2, n_3, n_4, n_5]\\ \textit{Ls} &= [n_6]\end{aligned}$$

Note that the order in which the new elements are pushed is not innocent of consequences. If we pushed them on in the reverse order then we would expand the VP first. Our choice of ordering will give us the effect of either a leftmost or rightmost derivation. Following the strategy here we get a top-down, left-to-right parser. This behavior should be familiar to students of context-free parsing. A pushdown automaton will be sensitive in exactly this way to which derived nodes are pushed onto its stack first.

Continuing with the example, we are now concerned to license n_7, at the top of the element-stack. Condition (33b) will apply, requiring in its environment the presence of a Det and an N. Once again, all of the elements already present in the description have category features other then those we are looking for, so we introduce another pair of new nodes, n_9 and n_{10}, once again pushing them onto the element-stack according to their precedence order.

(4.14)
$$\begin{aligned}\theta_2 &= \{X/n_7, Y/n_9, Z/n_{10}\}\\ \varepsilon_{3b}\theta_2 &= \{\textit{i-dom}(n_7, n_9), \textit{i-dom}(n_7, n_{10}),\\ &\quad \textit{i-prec}(n_9, n_{10}),\\ &\quad \textit{cat}(n_9) \approx \textit{det}, \textit{cat}(n_{10}) \approx n\}\\ \textit{EltList} &= [n_9, n_{10}, n_8, n_1, n_2, n_3, n_4, n_5]\\ \textit{Ls} &= [n_7, n_6]\end{aligned}$$

To license n_9 (the Det) we need to satisfy condition (33d). This time there are elements in the description which satisfy the required condition. In fact there are two of them: n_1 and n_4 are both of the required category. As a result we have actually three possible substitutions to consider: $\theta_3^1 = \{Y/n_1\}$, $\theta_3^2 = \{Y/n_4\}$, and instantiation as a new node: $\theta_3^3 = \{Y/n_{11}\}$.

Which substitution is attempted first will depend on the order of atoms in the working description, on the assumption that the parser scans the description from left to right, looking for substitutable instances. Thus it could happen that the parser would attempt the first substitution, $\{Y/n_1\}$, first. This would be the correct answer, but that result would be accidental. Indeed later on, when we are parsing the object NP, the correct answer will be n_4, so it is reasonable to suppose that the parser must guess wrong on one of these occasions. Without loss of generality, we consider the possible wrong choices here, rather than postpone the inevitable. Should the parser select θ_2^2 by mistake, we have the following scenario. Note first that there is no immediate problem with this choice. The resulting description is perfectly consistent. We will have to look ahead a little to see how the parser might cope with this problem. When the parser expands the VP, its head V will inherit the precedence properties of the VP (n_8), and so will provably follow the NP, n_7. Similarly, the Det of the NP (n_9) will inherit the precedence properties of the NP (n_7), and so provably precedes the V. Now, if the V dominates the terminal *kissed*, which is given in the input as preceding the second instance of *the* (n_4), then we have a crossing branch configuration. Exactly what reasoning leads to a contradiction depends somewhat on the implementation of the consistency checker. If it is set up to look for crossing branch configurations (which are somewhat complex, involving structural relations among four nodes), then it should be able to detect the problem at this point. Otherwise, it will continue to apply DownProp, ultimately giving that *kissed* both precedes and follows *the*. Such a configuration, violating the Exclusive Ordering theorem, involves only two nodes, and should be relatively inexpensive to detect. Otherwise, an instance of a reflexive predicate can be immediately derived, so we may assume that here at the latest the parser will realize that it has made an error, and it will begin to backtrack towards the correct substitution.

However, a serious problem arises here. Before going all the way back to consider the substitution it used to connect the subject Det to a supporting terminal, the parser will consider whether it did the right thing in attaching the V to the terminal *kissed*. Recall that here once again there is another possibility, namely that the V dominates a new, as yet unencountered node. Nothing I have said will prevent the parser from creating a fresh node, with category set to some verb in the lexicon (either *kissed* or *kicked*). Continuing in this way, the parser can 'hallucinate' an entire VP *after* the spurious NP [*the* ... *girl*].

One solution to this problem is to add to the parser a stipulation that it not be allowed to create new terminals. Accordingly, we could modify union* so that it knew which categories were terminals, and could check the features of any new nodes it created. Note that there is a very real sense in which the terminal elements are out of bounds, in fact, so such a move is not too unnatural. In any grammar consisting only of licensing conditions which license from below the terminal elements of a syntactic structure can never be licensed—there is nothing below them. So they really do seem to fall outside the parser's normal operation in any case. Assume then that this modification has been made to the parser.

Now the parser will not be able to find any alternative support for the V node besides node 3, and the crossing branch configuration will be inescapable. This will force the parser all the way back to the point at which it made the error. Note that now the third choice, the invention of a new node, is no longer available, since the new node has a terminal category.

(4.15)
$$\begin{aligned} \theta_3 &= \{X/n_9, Y/n_1\} \\ \varepsilon_{3d}\theta_3 &= \{\textit{i-dom}(n_9, n_1), \textit{cat}(n_1) \approx \textit{the}\} \\ \textit{EltList} &= [n_{10}, n_8, n_1, n_2, n_3, n_4, n_5] \\ \textit{Ls} &= [n_9, n_7, n_6] \end{aligned}$$

Of the statements in $\varepsilon_{3d}\theta_3$, only the dominance assertion is new, so it is the only one actually added to the working description. Also, no new nodes are added to the description in this step, so we move on to license n_{10}, the N.

The N node is licensed in much the same way as the Det node, except that the licensing condition for nouns has a disjunctive environment. In this case only one disjunct is consistent with the substitution θ_4, so this is the disjunct we pick. That is, if we tried the second disjunct first, we would be looking for a node Y with category *girl*. The only available node would be n_5. Pursuing this option would lead us to the same crossing branch configuration we encountered with the determiner nodes, and, backtracking, the parser would eventually settle on the correct noun.

(4.16)
$$\begin{aligned} \theta_4 &= \{X/n_{10}, Y/n_2\} \\ \varepsilon_{3e}\theta_4 &= \{\textit{i-dom}(n_{10}, n_2), \textit{cat}(n_2) \approx \textit{boy}\} \\ \textit{EltList} &= [n_8, n_1, n_2, n_3, n_4, n_5] \\ \textit{Ls} &= [n_{10}, n_9, n_7, n_6] \end{aligned}$$

Once again no new nodes were added, and only the dominance statement is new to the description.

The next node, n_8, is the VP. It introduces a V node and a new NP node. (Note that an initial attempt to use the already existing subject NP node n_7 as the object would once again lead to crossing branches.)

(4.17)
$$\begin{aligned} \theta_5 &= \{X/n_8, Y/n_{11}, Z/n_{12}\} \\ \varepsilon_{3c}\theta_5 &= \{\textit{i-dom}(n_8, n_{11}), \textit{i-dom}(n_8, n_{12}), \textit{i-prec}(n_{11}, n_{12}), \\ &\quad\ \textit{cat}(n_{11}) \approx v,\ \textit{cat}(n_{12}) \approx \textit{np}\} \\ \textit{EltList} &= [n_{11}, n_{12}, n_1, n_2, n_3, n_4, n_5] \\ \textit{Ls} &= [n_8, n_{10}, n_9, n_7, n_6] \end{aligned}$$

The V node, n_{11}, is licensed without introducing any new nodes. Once again the environment is disjunctive, but only one disjunct leads to a consistent description.

(4.18)
$$\begin{aligned} \theta_6 &= \{X/n_{11}, Y/n_3\} \\ \varepsilon_{3f}\theta_6 &= \{\textit{i-dom}(n_{11}, n_3), \textit{cat}(n_3) \approx \textit{kissed}\} \\ \textit{EltList} &= [n_{12}, n_1, n_2, n_3, n_4, n_5] \\ \textit{Ls} &= [n_{11}, n_8, n_{10}, n_9, n_7, n_6] \end{aligned}$$

The licensing of the object NP offers nothing new except that the parser will attempt to use the subject N and Det to license the object NP (just as it attempts to use the subject NP to license the VP node). This attempt will fail since the subject N and Det provably precede the object NP and so cannot be dominated by it as well. As a result the parser will have to introduce new nodes n_{13} and n_{14}, which will then be licensed pretty much as the subject Det and N were.

At this point we have all the structural information needed to pick out a unique tree. A certain amount of effort with the Transitivity and Downwards Propagation axioms will show that the description is exhaustively ordered. It is in addition rooted, and in all other ways consistent with $\mathcal{T}$. Therefore we ought to be done. However, we still have some unlicensed and unlicensable nodes in *EltList*. In particular, as observed above, no element of the input string will be licensed.

(4.19)
$$\textit{EltList} = [n_1, n_2, n_3, n_4, n_5]$$

However, since we had to give the parser knowledge of the difference between terminal and non-terminal categories, there is no reason not to take advantage of that distinction here. We simply redefine "element" to include only the non-terminal categories.

4.1.1 Avoiding some costly wrong moves

Throughout the preceding example parse, the parser regularly made wrong decisions. Though once it is suitably modified, it can find its way out of the traps set by its wrong decisions, nonetheless it clearly takes a good deal of reasoning to do so, and often the crucial inconsistent configuration only arises after a good deal of further parsing, meaning a deep search is wasted, and a correspondingly lengthy backtrack results. Obviously we would like the parser not to consider these erroneous moves in the first place.

Turning first to errors in selecting the correct terminal element, when the parser is looking for a terminal element to support a pre-terminal, it should be clear that there is really only one terminal which could possibly do the trick, namely, the leftmost one not yet attached into the structure. It would thus be extremely handy if the parser could know at any point which element that was.

The solution I will pursue here is as follows. First, we need to be able to treat the input as something with a fixed beginning and end, a sort of finite sequence. Accordingly I will assume that the function symbols of $\mathcal{L}$ include a quasi successor function, which I will call s, since it in many ways resembles the successor function of axiomatic arithmetic and number theory. For a sentence such as (20), we want the statements in (21) to be part of the theory at the start of the parse.[4]

(4.20) *The girl kicked the boy,*

(4.21)
$$\begin{array}{l} s(\mathit{first}) \approx n_1 \\ s(s(\mathit{first})) \approx n_2 \\ s(s(s(\mathit{first}))) \approx n_3 \\ s(s(s(s(\mathit{first})))) \approx n_4 \\ s(s(s(s(s(\mathit{first}))))) \approx n_5 \\ s(s(s(s(s(s(\mathit{first})))))) \approx \mathit{nil} \\ \mathit{cat}(n_1) \approx \mathit{the} \\ \quad\vdots \end{array}$$

In this example, the symbols *first* and *nil* represent special, silent sentence boundary symbols. They should denote individuals in the universe for which all functions are either undefined or return that individual (except of course that s is defined at *first* to be n_1, at least in this case). The latter is most

[4] Actually there are several equivalent representations. For example, it doesn't matter whether we call n_3 $s(s(n_1))$ or $s(n_2)$. Strictly speaking, it is sufficient that the statements in (21) be *theorems* of the initial state.

preferable, since we can continue to restrict ourselves to using $\approx$ as the only equality relation in logical descriptions.

It may appear at first that the s function is not significantly different from immediate precedence. Note however that a node can actually immediately precede a number of other nodes. For instance a node immediately precedes all of the nodes along the left edge of the 'next-door' constituent. On the other hand, because it is a function, $s(n)$ is unique for any given n. Thus the use of the s function is appropriate for defining something essentially one-dimensional, like a string. Going in the other direction is, however, perfectly possible—as long as we are talking about the input string (that is, the 'successors' of *first*) then succession does imply immediate precedence. Hence it may be redundant to include precedence information in the input description, as long as some principle tells us that succession implies precedence.

In addition, we add the function symbols *start* and *end*, which apply to non-terminals to yield that constituent's left edge, and the successor of its right edge, respectively. We thus need to modify the statements of the grammar along the following lines.

(4.22) CFLG_1 (Fragment)

(a) $\mathit{cat}(X) \approx \mathit{det} \Rightarrow$
$\{\mathit{i\text{-}dom}(X,Y), \mathit{cat}(Y) \approx \mathit{the},$
$\mathit{start}(X) \approx Y, \mathit{end}(X) \approx s(Y)\}$

(b) $\mathit{cat}(X) \approx \mathit{np} \Rightarrow$
$\{\mathit{i\text{-}dom}(X,Y), \mathit{i\text{-}dom}(X,Z), \mathit{i\text{-}prec}(Y,Z),$
$\mathit{cat}(Y) \approx \mathit{det}, \mathit{cat}(Z) \approx n,$
$\mathit{start}(X) \approx \mathit{start}(Y), \mathit{end}(X) \approx \mathit{end}(Z),$
$\mathit{end}(Y) \approx \mathit{start}(Z)\}$

(c) $\mathit{cat}(X) \approx s \Rightarrow$
$\{\mathit{i\text{-}dom}(X,Y), \mathit{i\text{-}dom}(X,Z), \mathit{i\text{-}prec}(Y,Z),$
$\mathit{cat}(Y) \approx \mathit{np}, \mathit{cat}(Z) \approx \mathit{vp},$
$\mathit{start}(X) \approx \mathit{start}(Y), \mathit{end}(X) \approx \mathit{end}(Z),$
$\mathit{end}(Y) \approx \mathit{start}(Z)\}$

The addition of *start* and *end* functions to the function symbols of $\mathcal{L}$ somewhat complicates the translation from CF productions to licensing conditions. Thus if we take once again a production like (4.23), below,

(4.23) $$A \longrightarrow B_1 \ldots B_k$$

then we need to add to its corresponding licensing condition the following extra sub-conditions.

(4.24) $$\begin{aligned}\{ & \quad \vdots \\ & start(X) \approx start(Y_1), end(X) \approx end(Y_k), \\ & start(Y_2) \approx end(Y_1), \ldots, start(Y_k) \approx end(Y_{k-1})\}\end{aligned}$$

In addition, for each B_i which is a terminal element we must replace all occurrences of $start(Y_i)$ which would be generated by (4.24) by Y_i, and all occurrences of $end(Y_i)$ by $s(Y_i)$.

Finally, when we add the root S node to the input description, we include the assertions:

(4.25) $$\begin{aligned} start(n_s) &\approx s(first) \\ end(n_s) &\approx nil \end{aligned}$$

where n_s stands for the root S node. This imposes the requirement on the root node that it dominate the entire input string.

The derivation of the initial Det will pass down the value of $start(S)$ so that all of the S node, subject NP and initial Det will have $s(first)$ as the value of their *start* features. Thus the Det already knows which terminal it wants to dominate, namely n_1, the successor of *first*, and there is no point in examining other instances of *the* elsewhere in the sentence. This orientation with respect to the input string is passed along, from the Det to the N, which establishes the end of the NP, and hence the start of the VP, which establishes the start of the V, and so forth.

4.1.2 Top-Down Parsing of Ungrammatical Sentences

The pointer-features introduced above also allow us to solve an otherwise extremely thorny problem for top-down licensing parsers. Consider the problem, not of parsing a grammatical sentence, but of parsing (or failing to parse) an ungrammatical one. The grammar interpreter of the last chapter, without pointer features, will function in such a way that many sentences which are ungrammatical with respect to a CFG will be accepted by the corresponding top-down CFLG. Consider example (26).

(4.26) **John slept the poodle.*

Assume first that CFLG_0 has been suitably modified to recognize that the category of *slept* is Intransitive Verb (IV), and that a VP can be licensed without an object if it dominates an IV. If we assume that terminal elements need not be licensed then ungrammatical sentences like (26) will always be accepted, since their extra words will never be involved in licensing relations.

Given the *start-end* features of the previous section, however, we have a neat solution available. Since the end of the S is *nil*, the VP will inherit this feature. Recognizing *slept* as an IV, the VP will be constrained to have its end equal the end of the V, which is the successor of *slept*. Thus the end of the VP equals both *nil* (by inheritance from above) and *the* (by inheritance from below). From this we derive a contradiction, since, assuming $cat(nil)$ equals *nil*, it cannot also equal *the*.

4.2 Licensing from Above and Bottom-Up Parsing

Just like Top-Down parsing with CFGs, a CFLG with the kind of left-to-right, top-down behavior induced by the stack regime on node selection and supported by the restriction of the grammar to licensing from below is susceptible to Left-Recursion-induced infinite searches. Consider the following left-recursive CFG.

(4.27) CFG_2 (A Left-Recursive Grammar)

(a) S $\longrightarrow$ NP VP

(b) NP $\longrightarrow$ *Waldo* | *Wilma* | NP Det N

(c) N $\longrightarrow$ *poodle* | *pangolin*

(d) Det $\longrightarrow$ *'s*

(e) VP $\longrightarrow$ *sneezed* | *slept*

Notice that an NP can be expanded to the sequence NP⌒Det⌒N. If we then proceed to expand the lower, possessor NP using the same production, we wind up with yet another NP to expand. There is no particular reason why we should ever stop expanding NPs as possessive constructions with new NPs. As a result the parser may never make contact with the input to find out whether or not the sentence really does begin with an NP. In such cases one remedy is to abandon a top-down approach for one which assembles the tree from the bottom up, so that it remains in contact with the data.

The immediate goal, then, is to convert the productions of CFG_2 into licensing conditions in a way compatible with bottom up parsing behavior. A standard CF parsing model with the desired bottom-up, left-recursion-defeating behavior is the Shift-Reduce parser, so we first consider how Shift-Reduce behavior might be induced in a licensing parser.

4.2.1 Shift-Reduce Licensing

The standard device in CF parsing for doing bottom up parses is the Shift-Reduce parser. Such a device works by turning the grammar on its head, so that it replaces right hand sides of CF productions with their left hand sides. If we attempt to treat shift-reduce parsing as licensing, we must treat the parser as licensing strings of categories, because this is what appears on the right hand side of context free productions. Unfortunately there is no way to write a licensing condition with the effect of a reduce operation, as long as we restrict ourselves to *node*-licensing—we would have to develop a more general logic for *configuration*-licensing to accomplish this. Such configuration-licensing conditions would, for example, place *three* nodes in the focus of the possessive NP condition. That is, the condition would read something like (4.28).

(4.28)
$$\begin{array}{l}\{cat(X) \approx np, cat(Y) \approx det,\ cat(Z) \approx cn,\\ end(X) \approx start(Y), end(Y) \approx start(Z)\}\\ \Rightarrow\\ \{cat(W) \approx np, i\text{-}dom(W,X), i\text{-}dom(W,Y), i\text{-}dom(W,Z),\\ i\text{-}prec(X,Y), i\text{-}prec(Y,Z),\\ start(W) \approx start(X),\ end(W) \approx end(Z)\}\end{array}$$

The key to defeating left-recursion effects is that, by moving the $X^\frown Y^\frown Z$ sequence into the trigger, we have forced it to be present already before its mother NP can appear—this is equivalent to the reason why Shift-Reduce parsing works, and doesn't run into left-recursion problems. Given a sentence like (29), below,

(4.29) *Waldo's pangolin coughed.*

once the lower NP has been projected from *Waldo*, there is nothing more that can be done with it, until we have projected *'s* and *pangolin*. Until then, the only sequence (in particular of triples) beginning with the NP would be $NP^\frown$*'s*$^\frown$*pangolin*, which presumably should not trigger any principled licensing condition. Only once we have the full sequence $NP^\frown Det^\frown N$ can we trigger the creation of the higher NP, and this in turn can only come about if elements in the input trigger the creation of the Det and N, from below.

If we pursue this approach to licensing from above, we would have to be very careful about stating the requirement of full licensing. Even if we restrict our attention to configurations which are start-end sequences, clearly not all such sequences in a tree will have applicable licensing conditions. Accordingly, we probably do not want to think of this scheme as licensing

sequences, but as licensing each of the nodes mentioned in the focus. This still could require some serious revisions of the parsing model of Chapter 3. In particular, if the order of licensing is controlled by recursion along the list of nodes, then we would have to have conditions like (4.28) apply three times, once as each node subject to its focus came under the parser's scrutiny. Otherwise, we would have to be able to add the name of conditions like (4.28) to the *Sat'd-Conds* list of each of the nodes satisfying its focus in one pass. Then, as the parser worked its way along the node-list, it would check, for each node it was attempting to license, whether a particular condition had already applied to it, and, if so, there would be no need to re-attempt the application of that condition to that node.

This is not really a problem for CFLGs, however. Since each node is licensed once, we can simply remove all three nodes from the unlicensed elements list as soon as condition (4.28) has applied to them. Let us, then, briefly sketch out a sample parse, using a CFLG corresponding to CFG_2.

First let us consider more generally how to cast a CFG as a 'sequence-licensing' CFLG from which we can get bottom-up parsing behavior with a suitably parameterized version of our parsing model. For productions of the form (30), below, where W is a terminal category, we have the licensing condition in (4.31).

(4.30) $A \longrightarrow W$

(4.31)
$$\begin{array}{l}cat(X) \approx w \qquad \Rightarrow\\ \{cat(Y) \approx a, i\text{-}dom(Y,X), start(Y) \approx X, end(Y) \approx s(X)\}\end{array}$$

For general productions of the form in (32), we have the corresponding licensing condition in (4.33).

(4.32) $A \longrightarrow B_1 \ldots\ B_k$

(4.33)
$$\begin{array}{l}\{cat(X_1) \approx b_1, \ldots, cat(X_k) \approx b_k,\\ end(X_1) \approx start(X_2), \ldots, end(X_{k-1}) \approx start(X_k)\}\\ \Rightarrow\\ \{cat(Y) \approx a, i\text{-}dom(Y,X_1), \ldots, i\text{-}dom(Y,X_k),\\ i\text{-}prec(X_1,X_2), \ldots, i\text{-}prec(X_{k-1},X_k),\\ start(Y) \approx start(X_1), end(Y) \approx end(X_k)\}\end{array}$$

Finally, where we have a production as in (34), where S is a start symbol of the grammar, we have, in addition to the condition generated as in (4.33), the root-licensing condition (4.35). The effect of the empty environment is to assure that any node meeting the conditions on X, i.e., which is an S which spans the entire input string, will be well-formed. Without such a condition no root node could be licensed in a grammar restricted to licensing from above.

(4.34) $S \longrightarrow B_1 \ldots B_k$

(4.35) $\{cat(X) \approx s, start(X) \approx s(first), end(X) \approx nil\} \Rightarrow \emptyset$

Accordingly, we have, for the grammar CFG_2, the corresponding CFLG, most of which is given in (36).

(4.36) *$CFLG_2$—A 'Shift-Reduce' Licensing Grammar*

(a) $cat(X) \approx waldo \Rightarrow$
$\{cat(Y) \approx np, i\text{-}dom(Y,X), start(Y) \approx X, end(Y) \approx s(X)\}$

(b) $cat(X) \approx {}'s \Rightarrow$
$\{cat(Y) \approx det, i\text{-}dom(Y,X), start(Y) \approx X, end(Y) \approx s(X)\}$

(c) $cat(X) \approx pangolin \Rightarrow$
$\{cat(Y) \approx n, i\text{-}dom(Y,X), start(Y) \approx X, end(Y) \approx s(X)\}$

(d) $cat(X) \approx coughed \Rightarrow$
$\{cat(Y) \approx v, i\text{-}dom(Y,X), start(Y) \approx X, end(Y) \approx s(X)\}$

(e) $\{cat(X_1) \approx np, cat(X_2) \approx det, cat(X_3) \approx n,$
$end(X_1) \approx start(X_2), end(X_2) \approx start(X_3)\}$
$\Rightarrow$
$\{cat(Y) \approx np, i\text{-}dom(Y,X_1), i\text{-}dom(Y,X_2), i\text{-}dom(Y,X_3),$
$i\text{-}prec(X_1,X_2), i\text{-}prec(X_2,X_3),$
$start(Y) \approx start(X_1), end(Y) \approx end(X_3)\}$

(f) $\{cat(X_1) \approx np, cat(X_3) \approx vp,$
$end(X_1) \approx start(X_2)\}$
$\Rightarrow$
$\{cat(Y) \approx s, start(Y) \approx start(X_1), end(Y) \approx end(X_3),$
$i\text{-}dom(Y,X_1), i\text{-}dom(Y,X_2), i\text{-}prec(X_1,X_2)\}$

(g) $cat(X) \approx v \Rightarrow$
$\{cat(Y) \approx vp, start(Y) \approx start(X), end(Y) \approx end(X)\}$

(h) $\{cat(X) \approx s, start(X) \approx s(first), end(X) \approx nil\} \Rightarrow \emptyset$

Let us now attempt to see how a licensing parser would parse the sentence in (29) using CFLG_2. The input would be given in the usual way, consisting of four nodes, $n_1, \ldots, n_4$, with categories appropriate to the input words. Going bottom-up, we do not have to add the matrix S node to the description, as we did going top-down.

It is easier to see the resemblance of the sequence-licensing parser to a shift-reduce parser if we divide the nodes into three lists, rather than the two we used with top-down licensing. That is, we have as before a list of licensed

nodes, but we divide the list of as-yet-unlicensed nodes in two, between the input nodes, and the fresh nodes added during the parse. I will write these two lists on the same line, with the new-node list being treated as a stack, read right-to-left. That is, the rightmost node is the top of the stack. So, given an illustration like (4.37),

(4.37) $[n_4, n_5, n_6] \qquad [n_1, n_2, n_3]$

we read the list on the right as the still-unlicensed nodes from the input description, and the list on the left as the nodes which the parser has created and not yet licensed. Of these, n_6 is meant to be the one most recently created, n_5 the next most recent, and n_4 the oldest. This display closely parallels the customary illustrations of shift-reduce parsers, with the stack on the left and the input buffer on the right. We will thus see quite easily that the behavior of the sequence-licensing parser closely parallels that of a shift-reduce parser.

Thus, initially, we have only the input nodes, as in (4.38).

(4.38) $[\,] \qquad [n_1, n_2, n_3, n_4]$

Node n_1 is subject to condition (3636a), which creates a new node n_5, an NP immediately and exclusively dominating *waldo*.

(4.39)
$$\begin{array}{rcl} \theta_1 & = & \{X/n_1, Y/n_5\} \\ \varepsilon_{36a} & = & \{cat(n_5) \approx np, i\text{-}dom(n_5,n_1), start(n_5) \approx n_1, \\ & & end(n_5) \approx n_2\} \\ [n_5] & & [n_2, n_3, n_4] \\ Ls & = & [n_1] \end{array}$$

The new node n_5 is potentially subject to licensing by either condition (3636e) or (3636f), except that there are no nodes in the description yet which have the right categories to fill out the triggers of these conditions. On the other hand, n_2 can trigger condition (3636b), causing the creation of a Det, n_6.

(4.40)
$$\begin{array}{rcl} \theta_2 & = & \{X/n_2, Y/n_6\} \\ \varepsilon_{36b} & = & \{cat(n_6) \approx det, i\text{-}dom(n_6,n_2), start(n_6) \approx n_2, \\ & & end(n_6) \approx n_3\} \\ [n_5, n_6] & & [n_3, n_4] \\ Ls & = & [n_1, n_2] \end{array}$$

We are still waiting for the handle of the NP-rule to develop on the stack, so the only thing to do is to license n_3, giving (41), below.

$$
\begin{array}{llll}
(4.41) & \theta_3 & = & \{X/n_3, Y/n_7\} \\
 & \varepsilon_{36c} & = & \{cat(n_7) \approx det, \textit{i-dom}(n_7, n_3), start(n_7) \approx n_3, \\
 & & & \quad end(n_7) \approx n_4\} \\
 & [n_5, n_6, n_7] & & [n_4] \\
 & Ls & = & [n_1, n_2, n_3]
\end{array}
$$

Now we finally have all the pieces we need to license the higher NP. The reader can easily verify that the *start* and *end* features of the three nodes on the stack do indeed coincide in the right way.

$$
\begin{array}{llll}
(4.42) & \theta_4 & = & \{X_1/n_5, X_2/n_6, X_3/n_7, Y/n_8\} \\
 & \varepsilon_{36e} & = & \{cat(n_8) \approx np, start(n_8) \approx n_1, end(n_8) \approx n_4, \\
 & & & \quad \textit{i-dom}(n_8, n_5), \textit{i-dom}(n_8, n_6), \textit{i-dom}(n_8, n_7), \\
 & & & \quad \textit{i-prec}(n_5, n_6), \textit{i-prec}(n_6, n_7)\} \\
 & [n_8] & & [n_4] \\
 & Ls & = & [n_1, n_2, n_3, n_5, n_6, n_7]
\end{array}
$$

Now the higher NP is subject to licensing by either (3636e) or (3636f). There are elements in the description with the right categories to trigger (3636e), namely the Det and the N just used to create n_8. However, the *start* and *end* features do not line up: the start of the Det (n_2) is not equal to the end of the higher NP (n_4). Therefore it cannot yet be licensed, and we turn our attention to *coughed*, n_4. I give the steps from here to the end of the parse below.

$$
\begin{array}{llll}
(4.43) & \theta_5 & = & \{X/n_4, Y/n_9\} \\
 & \varepsilon_{36d} & = & \{cat(n_9) \approx v, start(n_9) \approx n_4, end(n_9) \approx nil, \\
 & & & \quad \textit{i-dom}(n_9, n_4)\} \\
 & [n_8, n_9] & & [\,] \\
 & Ls & = & [n_1, n_2, n_3, n_5, n_6, n_7, n_4]
\end{array}
$$

$$
\begin{array}{llll}
(4.44) & \theta_6 & = & \{X/n_9, Y/n_{10}\} \\
 & \varepsilon_{36g} & = & \{cat(n_{10}) \approx vp, start(n_{10}) \approx n_4, end(n_{10}) \approx nil, \\
 & & & \quad \textit{i-dom}(n_{10}, n_9)\} \\
 & [n_8, n_{10}] & & [\,] \\
 & Ls & = & [n_1, n_2, n_3, n_5, n_6, n_7, n_4, n_9]
\end{array}
$$

$$
\begin{array}{llll}
(4.45) & \theta_7 & = & \{X_1/n_8, X_2/n_{10}, Y/n_{11}\} \\
 & \varepsilon_{36f} & = & \{cat(n_{11}) \approx s, start(n_{11}) \approx n_1, end(n_{11}) \approx nil, \\
 & & & \quad \textit{i-dom}(n_{11}, n_8), \textit{i-dom}(n_{11}, n_{10}), \textit{i-prec}(n_8, n_{10})\} \\
 & [n_{11}] & & [\,] \\
 & Ls & = & [n_1, n_2, n_3, n_5, n_6, n_7, n_4, n_9, n_8, n_{10}]
\end{array}
$$

$$
\begin{array}{llll}
(4.46) & \theta_8 & = & \{X/n_{11}\} \\
 & \varepsilon_{36h} & = & \emptyset \\
 & [\,] & & [\,] \\
 & Ls & = & [n_1, n_2, n_3, n_5, n_6, n_7, n_4, n_9, n_8, n_{10}, n_{11}]
\end{array}
$$

It should be fairly obvious, considering the dual lists in these displays, that the sequence-licensing parser is closely mimicking a shift-reduce parser, in particular one which prefers reduce to shift, since I have tacitly assumed that the parser searches the list of newly created nodes (i.e., the stack) before licensing a terminal. Note that licensing terminals actually combines into one step shifting a terminal onto the stack and reducing it to its covering preterminal.

In discussing this sequence-licensing strategy for bottom-up licensing I have sidestepped an important issue of control, which I will now attempt to face as squarely as possible. At various times in the preceding discussion I explicitly assumed that if the parser could not license some (sequence of) nonterminal node(s), it would ignore them and attempt to license a terminal. This behavior is not possible given the parsing logic presented in the last chapter. No matter what selection function is used to pick out a node to license next, once that node is selected, it is subjected to all the licensing conditions in the grammar, and if a condition fails to apply, it is discarded. If no condition applies, then the node is unlicensable, and the syntactic structure that includes it is ill-formed. No mechanism has yet been provide which would allow the parser to wait to license an NP, for example, until the crucial Det⌢N sequence had arisen alongside it.

The most promising approach to the control problem runs roughly as follows. The basic idea is that, instead of running through a list of nodes to license, and licensing or failing to license a single node at the single point in processing when that node comes up for examination, the parser is to run through a list of *tasks*. A task, in this sense, is a partially instantiated focus of a licensing condition. The parser must then cycle through this list of tasks looking for one which, given its current state of knowledge, it can fully instantiate. At that point it can determine if the nodes which are instantiated in focus of that task are licensed or not.

At the end of the parse, all uncompleted tasks are assumed to represent licensing conditions that did not in fact apply. It is only at the end of the parse that we can be certain that a node cannot be subject to a particular condition.

Referring to the example presented above, the first action of the parser would be to create tasks out of the input nodes. These would all be fully instantiated right from the start, since terminal nodes are subject to very simple licensing conditions. Once the NP over *waldo* was created in step one, it would itself trigger the creation of two tasks, corresponding to the two licensing conditions (3636e) and (3636f). Each would have to be tried, found uninstantiable, and returned to the task-pool. The first task would presumably be instantiated gradually, binding its instance of X_2 to the Det-node n_6 at an intermediate stage. The other task would never be further instantiated, and would still be in the pool when the parse was complete (i.e., when no more licensing was possible).

Clearly this strategy needs a good deal of rigorous development before it can be entertained seriously. However, it is interesting to note, even at this very sketchy stage, a strong resemblance between these tasks I have outlined here and the dotted rules of an Earley parser, or the pending arcs of an Active Chart parser.

4.2.2 Left Corner Licensing

We may be able to get bottom up behavior from a pure node-licensing grammar, where only a single node is properly seen as the subject of a licensing condition, but it takes a good deal of effort. The first move is to adopt the standpoint analogous to that which supported top-down behavior, and consider what happens if we take the grammar to consist of conditions restricted to 'licensing from above'.

If we take as a model a Left-Corner parser for CFGs,[5] rather than a Shift-Reduce parser, then we will see how this can be accomplished. Such a parser, in effect, goes bottom-up from the left corner, and then stops until each sister-node has been projected up from its own left corner, in the same way, before continuing with the mother constituent. In the case of our running example *Waldo's pangolin coughed*, we would project *Waldo* up to an NP and then project the Det and N as its sisters, pretty much as before, but instead of continuing on with the NP, we go bottom up from the next terminal, in an attempt to reach the Det, and then from the subsequent terminal in an attempt to build up to the N. Only at that point do we move up to the mother NP and begin looking for its required sisters.

Getting this behavior from a licensing grammar is a good deal trickier than getting Shift-Reduce behavior, and would require some modification of the interpreter. First, we will need to modify the form of the grammar, as expected, so that nodes are licensed from above. As an example, consider the condition on NPs, the focus of our problem with left-recursion, given in (47).

[5] Cf. [Pereira 1987, Aho 1972].

(4.47)
$$\begin{aligned}
&cat(X) \approx np \Rightarrow\\
&\quad \{i\text{-}dom(W,X),\ i\text{-}dom(W,Y),\ i\text{-}dom(W,Z),\\
&\quad\ \ i\text{-}prec(X,Y),\ i\text{-}prec(Y,Z),\\
&\quad\ \ cat(W) \approx np,\ cat(Y) \approx det,\ cat(Z) \approx cn,\\
&\quad\ \ start(X) \approx start(W),\ end(X) \approx start(Y),\ end(Y) \approx start(Z),\\
&\quad\ \ end(Z) \approx end(W)\}\\
&\vee \ \{cat(W) \approx s,\ cat(Y) \approx vp,\\
&\quad\ \ i\text{-}dom(W,X),\ i\text{-}dom(W,Y),\ i\text{-}prec(X,Y),\\
&\quad\ \ start(W) \approx start(X),\ end(X) \approx start(Y),\ end(Y) \approx end(W)\}
\end{aligned}$$

In addition, we will need to modify the way the parser chooses which node to license next, so that it will know when to work on a non-terminal, and when to look instead at the leftmost terminal. The alteration to the selection function is necessary to ensure that nodes are built up from below before they are licensed from above. The selection rule that I will impose on the interpreter is as follows.

(4.48) *Selection Rule for 'Left-Corner Licensing':*

Among the leftmost nodes, select the lowest.

The selection computation can be made easier by a careful implementation of the *start* and *end* functions. The parser should have already established, in the last paragraph's scenario, that the start of the Det is the end of the possessor NP, and that the end of the possessor NP is *'s*. It should be clear that a node will always dominate its start, and always (immediately) precede its end. Therefore whenever we are choosing between a non-terminal and a terminal, if the terminal is the start of the non-terminal, then it is the node to license next. If it is the end of the non-terminal, then the non-terminal is left of the terminal, and therefore it is the one to license next. Given the way in which starts and ends are passed among sisters, and between daughters and mothers, only these two cases should be possible. As noted, the correctness of this algorithm depends on the way in which the *start* and *end* functions are manipulated in a grammar. In particular the grammar must enforce that a node must share its *end* feature with its rightmost daughter, unless that daughter is a terminal node. In that case the mother must have its *end* feature equal to the successor of its daughter in the terminal string.

Once *'s* has been projected up to its Det and *pangolin* has been projected up to its N (assuming we are still trying to parse sentence (29)), then we must choose between licensing the higher NP or *coughed*. We can prove that the NP precedes *coughed* by noting that *coughed* is the end of the NP. In parsing the string *Waldo's pangolin coughed* we thus get the following order of licensing (using for convenience subscripted category labels, rather than node names).

(4.49) *waldo*, NP_1, *'s*, *Det*, *pangolin*, *N*, NP_2, *coughed*, *VP*, *S*

This is indeed a left-corner parse of the string *Waldo's pangolin coughed*, according to the definition in [Aho 1972, pp. 278-280]. The selection rule has the effect that a terminal will be licensed next unless the next available terminal is the value of *end* at the lowest leftmost non-terminal.

If a left-corner strategy can be made to work, it may provide a cleaner approach to bottom-up parsing with licensing grammars, since it does not get us into such difficulties with control of the parser as the sequence-licensing approach does. However, it should be noted that this strategy is not without difficulties.

4.2.3 Left Recursion Again

Despite the superficial resemblance to the licensing parser of the previous section to a left-corner parser, note that we have still not solved the problem of left recursion. Suppose we have a sentence like (50).

(4.50) *Waldo's pangolin coughed.*

We license *Waldo* by placing it in an NP, and then the NP by placing it under another NP, and following it with a Det and then a N. If we proceed to license the mother NP, of course, we are once again in a left-recursive situation, even though now we are building our potentially infinite stack of NPs from below rather than from above.

Unfortunately this remains true even if, following our latest strategy, we are careful to license the terminals *'s* and *pangolin* before we go on to license the NP, so as to lend support to the analysis of *Waldo* as a possessor. We are left with the upper NP and *coughed* as unlicensed nodes. Since *coughed* is the end of the NP, the NP will be licensed next. At this point nothing prevents the parser from trying the same analysis again, attempting to license the upper NP by including it as the possessor of a new, spurious NP. Because all licensing is from above, there is no need to support that analysis from the data. We will, of course, turn next to the terminal *coughed*, but that will not help. It will be projected up to a VP, eventually, and then to an S, looking for a subject NP. One possible substitution at this point is of the spurious NP with the subject of the VP, from which we will get that the end of the spurious NP is the same as the end of its possessor, *Waldo's pangolin*, but from this we can only conclude that the starts and ends of the spurious Det

and N are all the same, all equalling *coughed*. I have not done anything to rule out empty productions, as these are generally allowed in CFGs, and they would have exactly this result, that for any category expanded as the empty string, the start of that category will equal its end. So nothing will be seen as wrong with this structure, even though it clearly requires that Waldo's pangolin be the possessor of something unspecified.

We must also allow other substitutions, where the subject of the sentence is not equated with the spurious NP [*Waldo's pangolin Det N*]. In such cases the parser will be allowed to project up from the NP to yet another bad NP. This will take its turn as the subject, and then admit the possibility of yet a higher NP. Note that the problem here is not quite the classical problem posed by left recursive grammars, where no analysis is ever returned. Analyses are returned, infinitely many of them, in fact, and all wrong. Unless we attempt to license NPs as subjects before we try them as possessors this infinitely broad search will still block access to the correct result, and even if we do order subjects before possessors, we will still never finish finding all the possible analyses of the sentence, all but the first incorrect.

The problem is that, at the time we were licensing the VP, we should have been able to conclude that something was wrong. Only left corners introduce new structure in a correct parse. The sisters of the left corner are all licensed in an environment which has been built off of the left corner. Therefore the VP, a non-left-corner, should not have been allowed to build new structure, the NP and S nodes it requires in order to be licensed. As soon as such structure was required and found not to be present, the parser should have been able to conclude that something was wrong, and backtracked to the last point at which it had a choice, namely, the point at which it attempted to license [$_{NP}$ *Waldo's pangolin*].

Accordingly we might attempt the following revisions to the grammar interpreter: new nodes can only be added by licensing conditions whose triggers are the left corners of their licensing environments. Following such a strategy the VP node projected from *coughed* will be unlicensable (in particular in that it requires a missing mother S), and that erroneous branch of the parser's search space will be abandoned. This requires only that we mark licensing conditions according to whether they are allowed to introduce structure (a simple matter of inspecting the structural predicates in the environment to assure that the focus variable is the leftmost lowest node in that environment), and that we modify `union*` to require that the description only be altered if the environment is appropriately marked.

Note that we might expect some problems in dealing with categories which are sometimes left corners and sometimes not. For example, in a grammar with transitive verbs, NP objects are not left corners, even though NP subjects and possessors are. However, as long as we can mark the individual *disjuncts* of the licensing condition on NPs, and do not have to decide once and for all for the whole condition, then no problems arise. We might expect two sorts of misjudgments in the parser: if the NP is in fact a subject, say, and the parser tries to treat it as an object, and the opposite case, where an object is licensed as a subject.

The first error in the parser's judgment should be quickly detectable, since the required structure, namely the dominating VP and preceding V, will not already be present. The second error is slightly more pernicious: the object, misanalyzed as a subject, will be allowed to introduce new structure, an S and a VP. In this case the S so introduced will immediately dominate the misanalyzed NP. In addition, as the result of licensing the V of which that NP is the object, there will be in the description a VP node which immediately dominates the NP. This is inconsistent, and the parser will be able to rule out that analysis and proceed to the correct one. However, not all cases are so simple. If the NP were misanalyzed as a possessor we would have to wait until the Det so proposed failed to be supported by the input (that is, until the input projected something the licensing of which was blocked by the intervention of the spurious possessive construction, like the VP dominating *coughed* in our earlier example of left-recursion effects). The absence of a general result, and the dependence on projection from the input to be blocked, rather than for the inconsistency of the erroneous node itself to be recognized, remain somewhat troubling.

In the next section I will examine the behavior of this 'left-corner' licensing parser on ungrammatical inputs. We will see that this last problem falls together naturally with a problem in correctly handling too-short sentences, like **John hit*. Accordingly I will do no more than note the problem here, and move on to the discussion of bottom-up parsing of ungrammatical sentences.

4.2.4 Ungrammatical Sentences and Left Corner Licensing

Just as the top-down case presented problems with the recognition of ungrammatical sentences, so does the bottom-up case. Here the problems are more restricted, but they are also more intractable.

Sentences with extra words do not pose much of a problem here. Given a sentence like **Wilma's poodle coughed Waldo*, we get a number of conflicts. The sentence must end where the VP ends, and the VP must end where the intransitive V ends. If in addition the sentence must end at *nil*, then we get

the contradiction in (4.51) (abusing the notation slightly, for succinctness).

(4.51) $$end(S) \approx nil \wedge end(S) \approx s(coughed) \approx Waldo$$

However, sentences with missing words do pose a problem. Given a sentence like **Wilma's poodle bit*, and assuming a grammar with transitive Vs and VPs, then the parse will ultimately reach a point at which *bit* projects a V, and the V projects a VP and an NP. The lowest leftmost unlicensed node is then the NP, since we have run out of terminal string. Normally the next node would be a terminal, which we would then try to build up to the NP, but here no such element exists. The NP is then licensed from above by inclusion in the VP, and we proceed to a successful conclusion, wrongly. Note that the *start* and *end* functions and the constant *nil*, which have been so useful in the past, are of no use here. The situation is given in (52).

(4.52) (a) $end(V) \approx start(NP) \approx nil$,
(b) $end(VP) \approx end(NP) \approx nil$.

The first equality (5252a) is derived from a condition on transitive Vs and equivalently by a condition on NP objects, together with the description of the input (i.e., that $s(bit)$ is *nil*). The second equality (5252b) is derived from the condition I have assumed, that Ss end at *nil*, together with conditions on VPs, that their ends coincide with the end of their mother S, and again the condition on object NPs, that they end at the end of their mother VPs. The resulting conclusion, that the NP starts and ends at the same position, and that that position is *nil*, is entirely consistent with any condition we could reasonably put on a licensing-from-above LG. (It is in fact just the situation which would arise if we were to try to model CFGs with empty productions.) Once again, this is probably a good analysis of the bad sentence, but the parser has no way of supporting an intuition that the sentence was bad.

4.3 Combining strategies

At this point it is time to re-examine the goals of maintaining a distinction between licensing from below and from above. It should be clear by now that we can get very close to a top-down parser using only licensing from below, and we can get fairly close to bottom-up parsing behavior using only licensing from above. If we consider the way in which a CFG 'licenses' nodes by rewriting expansions, it should be no surprise that we are closer to success with a strictly top-down, licensing-from-below grammar. It is normally the case in CF parsing that bottom-up behavior is gotten either by transforming the grammar off-line or by being able to inspect the structure of the rules on-line. In our case we could presumably do such a thing by altering the grammar interpreter, and it is likely that many of the problems in getting bottom up parsing from the licensing logic of the previous chapter stem from my reluctance to significantly alter the interpreter here.

Furthermore, the problems we encounter studying top-down and bottom-up licensing parsers all stem from the restriction to half of this licensing picture. We have trouble licensing terminals from below and roots from above, and we have great difficulty preventing hallucination in a bottom-up parser simply because it is prevented from checking incorrectly hypothesized constituents against the input string. If, in the left-recursive case, we require that the spurious Det and N be licensed from below as well as from above, we will find that they are not so licensed, that there is no lexical item in the right place in the input to support either element.

Accordingly I will consider now grammars which are the union of licensing-from-below with licensing-from-above interpretations of a CFG. Combining licensing conditions does not affect the directional behavior of a licensing parser, since this is a function primarily of its selection operation. Nonetheless, if we use a stack-based, LIFO selection function, we find that only the licensing-from-below conditions actually contribute new information to the description. That is, supposing we start by licensing the root S, this creates new NP and VP nodes. Licensing the NP next, it is licensed from above by inclusion in an S, a structural configuration which has already been established. Only when we license it from below do we actually create new nodes (the Det and the N, in this case). This should actually be intuitively obvious from the (near-) correctness of the parser given only a grammar of conditions which license from below. Similarly, if we use a lowest-among-the-leftmost selection strategy we find that only the licensing-from-above conditions actually contribute new information. In fact, as already noted, only those conditions which apply to left corners actually contribute new information. The redundant licensing-from-below conditions are normally satisfied by structure already built by the licensing-from-above/bottom-up conditions.

The redundant conditions (the bottom-up conditions under a top-down selection function, and the top-down conditions under a bottom-up selection function) do play a role when it comes to ruling out ungrammatical sentences. Taking as an example the last, hardest problem, of missing words in a bottom-up implementation, we can see that this would not pose a problem if the spurious object NP had to be licensed from below as well as from above. If the relevant terminals had been present in the string then the appropriate licensing configuration supporting the NP from below would already have been built by the time the NP had to be licensed. In the absence of these terminals, however, the NP will cause its supporting structure to be built and, ultimately, checked against the input, causing the sentence to be rejected. Note, in passing, that this more powerful, directionless grammar is not powerful enough to obviate the need for a sequence representation of the input. In the case just discussed, if the input were not provided with a definite beginning and end, then the top-down licensing of the NP would still hallucinate the relevant supporting lexical items.

However, this combined grammar does present a difficulty in dealing with precisely such cases as a spurious trailing NP. Because it allows licensing from below, and supposing that the grammar is left recursive, matching that NP up against the input could be quite time-consuming! However, our last step towards a solution of the bottom up parsing problem is useful here too. There we noted that, since structure should only be built by a certain subset of the licensing conditions, we could consider modifying the interpreter slightly to encode this observation as a requirement. We could do the same here with the conditions that license from below. Assuming we still have a leftmost-lowest selection rule, then, we could relegate these conditions to a *structure checking*, rather than structure building role. This blunts their tendency towards infinite search by limiting their field of vision to one level of structure. In the case of the trailing object NP node in **Wilma's poodle bit*, the NP cannot find its supporting structure, and the parser will accordingly realize its error. On the other hand, given the analogous correct sentence *Wilma's poodle bit Waldo's pangolin*, the leftmost-lowest selection function, now provided with actual lexical material to work on, will guarantee that the terminal element *waldo* will be licensed before the object NP, once again holding in check its left recursive proclivities. When the parser does check the NP to see that it is properly supported, that support will already be in place, as required.

The same approach offers a way of handling bad parses of good sentences, as in the case of *Waldo's pangolin coughed*. There, if the parser attempts to analyze $[_{NP}$ *Waldo's pangolin* $]$ as a possessor, the spurious Det so introduced will fail the structure-checking condition attempting to license it from below, in this case immediately, since Dets are in direct contact with the input.

I have observed in various places in this chapter that the behavior of the parser on ungrammatical sentences is very close to what we would want it to do, if only it could still recognize the fact that it had been given a bad input. It seems possible that an implementation could be developed, departing somewhat from the strict logic of licensing and the format of licensing conditions outlined in previous chapters, which took advantage of this separation between conditions which build structure and conditions which check structure to achieve this goal of efficient parsing without sacrificing sensitivity to grammaticality. In such a development, the primary, structure building conditions could be employed to develop a fast sketch of the syntactic structure, while the secondary, structure checking conditions could be employed in grammaticality judgment, and in parsing in the way already outlined, to steer the primary parser away from bad structures when good structures were still an option. By carefully switching control between the primary and secondary routines (favoring the primary ones sometimes more, sometimes less), we might be able to model various kinds of parsing errors, in particular as they are induced by varying amounts of pressure on the parser's time and space resources. This is a question I will leave open here, in particular as I have not yet established that any comparable redundancies exist in grammars other than context-free. However, it seems an avenue of potential interest for research in cognitive modeling.

Chapter 5

From Phrase-Structure Grammars to Principle-Based Grammars

In the previous chapter we noted that the translation of a context-free grammar into the Licensing Grammar formalism yields an object which is considerably less concise than the source CFG (refer to the discussion on page 75). Various specific reasons were advanced there for the extra explicitness that is required in licensing grammars. The central thesis of this work is that this extra explicitness is highly desirable when we move beyond phrase-structure grammars to principle-based grammars. More specifically, the features which allow for the conciseness with which context-free grammars can be expressed in a rewrite formalism are lacking in principle-based grammars.

Before launching into an investigation of principle-based grammars, let us try to elucidate various assumptions which are implicit in the string rewrite formalism, and which we can relax in the licensing-based formalism developed here. As we relax the various features which distinguish context free licensing grammars from more general licensing grammars we will begin to see a more or less continuous path from phrase-structure grammars to principle-based grammars.

There are three features in particular which distinguish the kinds of statements in which we encoded CFGs in the last chapter from the kinds of statements we will need to approach GB Theory.

1. Expansion is *downwards*—Nodes are expanded into their daughters, exclusively.

2. Expansion is *complete* and *unique*—Nodes are expanded at most once; any structure which a particular node introduces is all introduced in a

single rule application.

3. Expansion is *strictly local*—Nodes are only expanded by introducing other nodes immediately adjacent to them.

The first of these conditions is the easiest to relax. Indeed we have already seen an example, in a sense. If we look at the way in which a left-corner parser interprets a context-free grammar, we can think of it as expanding a left corner upwards (and rightward). All of the other conditions remain true of such a system. Pursuing familiar examples, if we consider ID/LP ('immediate dominance/linear precedence') grammars, we can consider them examples of the relaxation of the second condition, leaving the other conditions (more or less) intact. Such grammars (GPSG and HPSG are the most familiar examples) include a system of rewrite rules of the familiar sort which, however, are interpreted only as expressing dominance relations. That is, a rule like (1), below, is interpreted as saying that S dominates NP and VP, but no commitment is made as to the precedence order of NP and VP with respect to each other. Linear precedence is fixed by a different set of rules, whose domain of application *overlaps* that of immediate dominance rules like (1).

(5.1) S $\longrightarrow$ NP VP

We will shortly see, as we turn our attention to GB Theory, that we must relax all of these conditions at one time or another. The focus of this chapter will be on the characterization of several configurations which are important in the theory, and whose characterization requires us to use the full resources of $\mathcal{L}$. The chief configurations I will present will be the sisterhood configuration of Theta Theory, the head-government configuration of Case Theory, and the antecedent-government configuration of Chain Theory, together with the organization of nodes into $\bar{\text{X}}$ projections and chains.

I will not attempt a full exposition of GB Theory, nor even of the modules which I will touch on. Thus the collection of licensing conditions which I will present here should not be taken as a theory of natural language, nor even of a fragment of natural language. It is a feature of principle-based grammars, where structure emerges through the interaction of very general and widely applicable principles, that, though very complex constructions depend on a remarkably small number of principles, very simple constructions do not depend on an appreciably smaller number of principles. That is, it is just about as difficult to specify the properties of a simple sentence as those of a complex one. In any case the goal of this work is not to recast a particular strand of GB as a parser, which is the tack taken by the majority of Principle-Based Parsing work, but to explore more general issues that arise when we

look at Principle-Based Grammars as Licensing Grammars. As a result I will frequently revise the various conditions I propose here, and will feel free to offer alternative realizations of key theoretical concepts without attempting to offer any empirical arguments either way. The spotlight here is intended to be more on Licensing Grammars than on (any particular variant of) GB Theory.

In the discussion that follows I will make a number of simplifying assumptions. In particular, I will restrict my examination of movement phenomena just to instances of A-chains of maximal projections. Furthermore, I will not attempt a multi-stratal account here—defining relations among different levels of representation would require more formal apparatus than I have developed. Hence I will be treating movement as chain-formation, and will pay careful attention to constructing versions of $\bar{\text{X}}$ Theory and Theta Theory which can apply at s-structure.

5.1 Argument Structure and $\bar{\text{X}}$ Theory

5.1.1 Theta Theory and Argument Structure Projection

We turn our attention first to Theta Theory, the module of GB Theory which comes closest to doing the work of a phrase structure grammar. The essence of the theory involves first the notion that all lexical items are given in the lexicon with an argument structure *grid* (to use the terminology of [Stowell 1981]). Lexical heads are construed as predicates, and their grids express the particular constraints which these heads place on the phrases which are to be interpreted as their logical arguments.

In this section I will discuss two different encodings of the grid (or θ-grid, as it is often called). In both cases the structure we are interested in is a set or sequence of *slots*, which will be identified with the arguments which we would say fill those slots in the logical interpretation of the sentence. Thus a verb like *buy* would include in its grid slots corresponding to the phrases which identify the one who buys (the *agent*, or perhaps *goal*), the person who is bought from (the *source*), the thing bought (the *theme*), and, perhaps, the *amount* for which the exchange is made. The verb *buy* places certain restrictions on what can fill each of these slots. As far as the syntax is concerned, the phrases which fill the slots of the grid, to which these thematic roles are assigned, will have to have certain categorical features. Each grid position is thus associated with a subset of the features which characterize the phrases which will ultimately be associated with these slots. In addition, grid positions must be marked according to whether they are *obligatory* or *optional*. Finally, one slot is distinguished as the *external* argument, which is usually thought of as the logical subject.

The grid is usually thought of in the literature as an unordered set of such slots. However, it is beyond the power of logical descriptions to characterize sets. Some sort of ordering must be imposed on arguments in order to express them as part of a logical description. As it happens, it is generally felt that there is an ordering underlying the assignment of thematic roles with respect to a grid. This *thematic hierarchy* is taken to dictate structural relations among a predicate's arguments so that, for example, a theme will be attached lower in the predicate phrase than a source or a goal. In such a case we can make of necessity a virtue and cast the logical description of a grid as a sequence where the order of elements corresponds to their relative thematic prominence.

Thus suppose we have a node n_1. One way of describing its argument structure grid is with a feature $grid(n_1)$. Then we can denote the first, most internal argument of this grid as $arg1(grid(n_1))$. Using the function s to suggest the successor function of axiomatic arithmetic, the second argument would then be $s(arg1(grid(n_1)))$, and the third would be $s(s(arg1(grid(n_1))))$, and so on. Note, however, that the function *grid* is invariably composed with *arg1*, so I will generally dispense with an explicit *grid* feature, understanding the argument structure grid as *arg1* and its successors.

We can assign relevant properties to each position with our usual equational descriptions of feature assignments.

(5.2) $role(arg1(n_1)) \approx theme$
$obliq(arg1(n_1)) \approx +$
$external(arg1(n_1)) \approx -$

We can then express thematic role assignment by identifying phrasal nodes with positions in a predicate's grid, as in (3). Here we assume that n_2 is a phrasal node in the proper structural relation to our thematic role assigning predicate, n_1.

(5.3) $arg1(n_1) \approx n_2$

This assignment has the immediate consequence in (4).

(5.4) $role(n_2) \approx theme$

If, in addition, n_1 places categorial restrictions on its arguments, then the identification in (3) acts to impose these restrictions at the same time as it

supplies new information about n_2 (i.e., its thematic role). Suppose, for now, that we treat category as a feature applying directly to a node, so that we might say $cat(n_2) \approx np$. We apply this same function to the grid position $arg1(n_1)$, giving (5).

(5.5) $cat(arg1(n_1)) \approx np$

Now suppose we have in addition a node n_3 whose category is clause, or *cp*, in current terms. If we attempt, erroneously, to associate n_3 with the first argument of n_1, we will derive the equation $np \approx cp$, which is inconsistent with the axioms of equality, in particular $\mathcal{T}191$, on page 32. Only if we assign the first argument slot of n_1 to a noun phrase can we wind up with a consistent theory.

Two principles underlie most accounts of Theta Theory. The first is the Projection Principle, which holds that all obligatory grid positions must be realized in the syntax, at all levels of structure. The second is the Theta Criterion, which holds that all argument-type categories (which we can think of for our present purposes as NPs) must receive a thematic role from some predicate. These two conditions are almost, but not quite, symmetric. There can be more NPs in a structure than there are obligatory slots, though there cannot be more NPs than there are slots, nor can there be fewer NPs than there are obligatory slots.

The Projection Principle can be expressed as in (6). I have given a version of the Theta Criterion in (7).

(5.6) *Projection Principle*
$oblig(arg1(X)) \approx + \Rightarrow$
$\{node(V1), V1 \approx arg1(X)\}$

(5.7) *Theta Criterion*
$cat(X) \approx np \Rightarrow$
$\{node(V1), X \approx arg1(V1)\}$

The version of the Projection Principle which I have given above as (6) is triggered by the presence of an argument slot which is [$+obligatory$]. All obligatory arguments will trigger it, forcing the existence of some node assignable to $V1$. If this node is not then fully licensed, the structure will be ill-formed, ultimately because some obligatory thematic position went unfilled.

Note that θ-role assignment is meant to be *bi-unique*. That is, every NP receives exactly one θ-role, and every θ-assigner assigns that role to a unique NP, if it assigns the role at all. As it happens, the structural conditions on θ-role assignment are sufficiently strict to rule out any violations of this biuniqueness condition. However, if some stronger mechanism is required, note that the characterization of θ-role assignment via the formula $V1 \approx arg1(X)$ already guarantees that a given role can be assigned to only one argument. This is because *arg1* is a function, and must denote a mapping from unique X to unique argument of X. The axioms governing equality will assure us that if $V1 \approx arg1(X)$ and $V2 \approx arg1(X)$ both hold, then so does $V1 \approx V2$, by the substitution of equals for equals. The same device can be used in reverse to get us from uniqueness to biuniqueness. We need only add as part of Theta Theory, say in the Theta Criterion, that not only does $X \approx arg1(V1)$, but also $V1 \approx \theta\text{-}assigner(X)$. Then by the above reasoning, if $V1$ is assigned roles by (apparently) multiple predicates, those predicates will be provably equal.

This identification is restricted by several conditions, in the version I present here. First, grid positions can only be discharged to *strict sisters* of the predicate. This condition on θ-role assignment is quite common in the literature, but the interpretation of 'sisters' can vary a good deal. In this first version of Theta Theory I will adhere to a classical definition of sisterhood: two nodes are sisters if they are immediately dominated by the same node. Below, in §5.1.3, I will consider what would have to be done to allow for looser versions of the sisterhood configuration (as it appears, for instance, in [Chomsky 1986b]).

(5.8) *Sisterhood Condition*
$X \approx arg1(Y) \Rightarrow$
$\{node(V1), i\text{-}dom(V1, X), i\text{-}dom(V1, Y)\}$

All that the mechanism introduced so far can do is to discharge the first argument slot of a predicate's grid. One way to deal with multiple arguments is essentially to follow the lead of Categorial Grammar. According to this scheme an n-place predicate combines with one of its arguments to yield an $(n-1)$-place predicate, which then combines with another argument and so forth until the predicate is saturated. We can implement this essentially binary-branching scheme as part of Theta Theory quite simply, by assigning to the mother of a θ-assigner the remainder of its daughter's grid, as in the Grid Projection Condition, below.

(5.9) *Grid Projection Condition*
$X \approx arg1(Y) \Rightarrow$
$\{i\text{-}dom(V1, Y), arg1(V1) \approx s(arg1(Y))\}$

Note that, given the way the parser interprets licensing conditions, there is no way to stop this process of projecting grid-remainders up the tree. Suppose we have a given node α, which assigns to its sister the very outermost argument of its lexical head. Suppose further that we have a node

β, the mother of α. The Grid Projection Condition will then require that $arg1(\beta)$ equals $s(arg1(alpha))$. The parsing model of Chapter 3 will obediently assert the necessary equation, thereby calling into existence an element $s(arg1(alpha))$, for which there is no lexical support.

Suppose, then, that we require in the focus of the Grid Projection Condition that $s(arg1(X))$ be already defined before we project it as *arg1* of the mother of X. This works because we require licensing conditions to apply only to elements which already exist in the description; this is built into the definition of the predicate applies_to of Chapter 3.

(5.10) *Grid Projection Condition 2*
$s(arg1(X)) \approx Y \Rightarrow$
$\{i\text{-}dom(V1, X), arg1(V1) \approx s(arg1(X))\}$

Note that in this example we have no interest at all in the value of Y. The expression $s(arg1(X)) \approx Y$ can really be taken to mean "the function $s(arg1(X))$ is defined in the current description." In what follows, where the appearance of a variable is purely formal, and no interest attaches to its value I will simply write '__'. This symbol should be interpreted as generating a fresh, arbitrary variable every time it appears.

In addition to these conditions, θ-role assignment is required to apply only to maximal projections, and it is subject, at least in some theories, to a directionality parameter (cf. [Koopman 1984, Travis 1984]). In English it is generally assumed that this parameter causes θ-roles to be assigned rightward, with the exception that the external θ-role is assigned leftwards. In addition the external role is generally held to be assigned by the maximal phrasal projection. I will ignore these details here, and treat all assignment as to the right, and I will treat the external grid slot as just the highest of the slots, discharged in every other way parallel to them.

(5.11) X^{max} *Condition*
$X \approx arg1(Y) \Rightarrow$
$\{X \approx max(proj(X))\}$

(5.12) *Directionality Parameter (English)*
$X \approx arg1(Y) \Rightarrow$
$\{i\text{-}prec(Y, X)\}$

The notation $max(proj(X))$ for maximal projection will be explained momentarily. It is properly part of the vocabulary of the $\bar{\text{X}}$ Module.

Taken as a whole, this version of Theta Theory comes very close to treating the θ-grid as a collection of quasi phrase structure rules, expanding the θ-assigning predicate into its immediate neighbors. Expansion, in this sense, is strictly local, by our classical interpretation of strict sisterhood, and it is meant to be complete. That is, the structure which should arise as a result of θ-assignment as it is specified here should be a binary branching tree, so that every daughter is either a θ-assigner or its argument, and every mother is the inheritor of the remainder of its daughter's grid.

If we take the conditions on θ-assignment individually we find that they are not in fact complete. The Sisterhood Condition is like the Immediate Dominance rules of an ID/LP grammar, with the Directionality Parameter acting as the (sole) LP rule. This decomposition is computationally artificial, of course, since each of these conditions share equivalent foci, and so they could be combined into a single licensing condition. However, since each of these conditions is conceptually distinct we might argue that they should be treated as separate statements in the grammar.

Though the combined effect of the conditions on θ-assignment is certainly complete with respect to the *structural* relations among nodes, there are of course various *features* of the nodes which take part in predicate/argument relations which are completely unspecified by Theta Theory. That is, Theta Theory is sensitive only to the presence of argument structure grid features on an element. Its particular category features, for example, are completely irrelevant. Theta Theory will treat alike verbs, adjectives, nouns and prepositions, as long as their argument taking properties are alike. However, setting aside features and focussing on structure, for the moment, we note that the licensing conditions which make up Theta Theory are still very close to having the characteristics of phrase structure rules: the conditions on θ-assignment are local and complete, though they are certainly not downwards.

5.1.2 $\bar{\mathrm{X}}$ Theory and $\bar{\mathrm{X}}$ projections

$\bar{\mathrm{X}}$ Theory in some sense evolved out of the phrase structure rules of the Base in earlier versions of Transformational Grammar. As a result it is often (erroneously, I think) taken to be the module of GB Theory which preserves the vestiges of a phrase structure grammar. As I present it here, $\bar{\mathrm{X}}$ Theory actually contributes very little to the structure of a phrase, being mostly concerned with regulating the spread of features throughout a phrase. (This version of $\bar{\mathrm{X}}$ Theory is perhaps closest in spirit to the treatment in [Speas 1991].)

The key concept of the $\bar{\mathrm{X}}$ module is that of the *projection*. To distinguish this somewhat overused term from the sense in which it is used in the Projection Principle (6, above), I will try to refer to 'projection' as it is used in $\bar{\mathrm{X}}$ Theory as *$\bar{X}$ projection*, as distinct from *Argument-Structure projection*.

An $\bar{\mathrm{X}}$ projection is a collection of nodes all of whom depend for their

category features on the properties of a *head*, a lexical-level node whose category features are an inherent property given to it in the lexicon. This dependence can be expressed several ways. Most often it is expressed through a chain of local dependencies, so that every non-lexical node is required to have among its immediate daughters a single distinguished daughter from which it inherits its categorial specification. Another way to express this dependence (implicit in the language I have been using so far) is to treat the members of a projection as elements of a class of a certain sort, so that the class as a whole has a distinguished element (the lexical head) which assigns its features to all the members of its $\bar{\mathrm{X}}$ projection.

Accordingly, I will assume that we have in $\mathcal{L}$ a feature *proj*, so that $proj(X)$ will be taken to denote the $\bar{\mathrm{X}}$ projection of which X is a member. Then the formula $proj(X) \approx proj(Y)$ will be taken to mean that X and Y are members of the same projection. The head of a projection can be written as $head(proj(X))$. Thus, if

$$proj(X) \approx proj(Y),$$

then

$$head(proj(X)) \approx head(proj(Y)).$$

We could go on to say that if X and Y are in the same projection, then they must share the same category features. Note, however, that the class of features shared among projection members is perhaps more general than just category features. Thus if a lexical item refers to a third-person entity (e.g., *boy*, or *statement*), then the phrase it heads will be interpreted as third person as well. Also, if accusative case is assigned to a phrase, in languages with overt, morphological case-marking, then that case will be marked on the head of the phrase. So there appears to be a collection of features of which category features like *noun* or *verb* are only a part. I will refer to these features as the *core features* of a node, writing $core(X)$. Accordingly we have the Endocentricity Condition, given below.

(5.13) *Endocentricity Condition*
$proj(X) \approx \_\_ \Rightarrow$
$\{core(X) \approx core(head(proj(X)))\}$

This has as a consequence that, if

$$proj(X) \approx proj(Y),$$

then

$$core(X) \approx core(Y).$$

Note that the introduction of *core* features requires some slight modifications to Theta Theory. In particular if θ-role assignment is the identification of a node with a slot, then we must specify the category selection features of, e.g., $arg1(n)$ as the value of a feature $cat(core(arg1(n)))$, rather than simply $cat(arg1(n))$. This slight modification guarantees us that all conditions which a predicate places on its arguments are imposed via the single equation $arg1(X) \approx Y$. That is, whatever feature structure we decide to assign to nodes, we must mimic that structure in the features we attach to grid positions.

Aside from heads, $\bar{\text{X}}$ projections also have maximal elements, which we can write as $max(proj(X))$. The number of intermediate (non-minimal, non-maximal) nodes in a projection depends on the number of phrases attached into that projection as sisters (e.g., as arguments or modifiers). However, every $\bar{\text{X}}$ projection will have a maximal element and a head element.[1] In addition, the external syntax of maximal projections is determined by their heads in more ways than just the sharing of category, or inherent features. Accordingly, we have the following X^{max} Condition.

(5.14) X^{max} *Condition*
$lexeme(X) \approx \_ \Rightarrow$
$\{node(V1), V1 \approx max(proj(X)),$
$proj(V1) \approx proj(X), dom(V1, X),$
$lexical(V1) \approx +\}$

The lexical status of maximal projections is relevant to Case Theory and to Chain Theory, which I will discuss below.

The notion of maximal projections suggests further slight revisions of Theta Theory, since now there are no longer categories like *np*. Rather, we have maximal elements of projections headed by nouns, and their category features will be identical to those of their heads. Accordingly we would have to replace the focus of the Theta Criterion, for example, with an expression like the following.

$$cat(core(X)) \approx noun, X \approx max(proj(Y)) \Rightarrow \ldots$$

[1] This is true for d-structures, where $\bar{\text{X}}$ Theory is normally considered to hold. Below I will consider the consequences of $\bar{\text{X}}$ Theory for s-structures of a simple sort, and there, where some phrases are traces of movement, it may not always be the case that every projection has a head. In particular, we might think of traces as being a one-element projection, of which the trace is the maximal element, as long as we are talking about the traces of maximal categories. I will return to this in more detail when we consider Chain Theory and Trace Theory, below.

We also need conditions to guarantee that if a node is a maximal projection then it is indeed maximal in its projection, and similarly that heads are minimal.

(5.15) $X \approx max(proj(Y)) \Rightarrow$
$\{X \approx Y\}$
$\vee$
$\{dom(X, Y)\}$

(5.16) $X \approx head(proj(Y)) \Rightarrow$
$\{X \approx Y\}$
$\vee$
$\{dom(Y, X)\}$

In addition, nothing yet guarantees that the head or maximal projection is actually a member of the projection.

(5.17) $X \approx max(proj(Y)) \Rightarrow$
$proj(X) \approx proj(Y)$
$X \approx head(proj(Y)) \Rightarrow$
$proj(X) \approx proj(Y)$

Finally, it should be noted that nothing I have said so far guarantees that a projection will be *continuous*. We should require that there exist a dominance-chain, from the maximal projection to the head, which is not interrupted by any node which is not a member of the projection. Put another way, we can state logically that every node is either a maximal projection or it is immediately dominated by a member of its projection, and in addition, every node is either the head of a projection, or it immediately dominates a member of its projection. Given the existence of definite endpoints of this chain, in fact only one of these conditions is necessary, so that we have the following.

(5.18) $\bar{X}$ *Continuity Condition*
$node(X) \Rightarrow$
$\{X \approx max(proj(X))\}$
$\vee$
$\{i\text{-}dom(V1, X), proj(X) \approx proj(V1)\}$

The Continuity Condition is problematic for the following reason. Each application of the condition might create a new node, which would be subject to licensing by the Continuity Condition in its own turn. Therefore it would always be an option for a projection to be infinite, even though bounded from above and below. The implementation of continuity in (18) actually is not subject to this infinite regress. Its output does not feed new applications of the condition for the simple reason that, though it creates objects that we intuitively think of as nodes (since they take part in dominance relations, and belong to projections), it does not claim of the objects it creates that the predicate *node* is in fact true of them. Thus only when $V1$ of the example is identified with a node whose existence (and node-hood) is vouched for by some other condition will it be subject to further licensing by this condition. This dodge works, apparently, but it is not conceptually particularly appealing. It may be better to assure that all conditions which attach phrases into other phrases (Theta Theory is the only example of this which I will consider, but Modification, and movement by adjunction are also examples) perform this attachment in such a way as to recognize the sister of the attached phrase and its immediate mother, and to assure that these phrases are labeled as elements of the same projection. Thus for example we would alter the Grid Projection Condition (10, above) by adding such a condition, as in (19).

(5.19) *Grid Projection Condition 3*
$s(\mathit{arg1}(X)) \approx Y \Rightarrow$
$\{\mathit{i\text{-}dom}(V1, X), \mathit{arg1}(\mathit{grid}(V1)) \approx s(\mathit{arg1}(\mathit{grid}(X))),$
$\mathit{proj}(V1) \approx \mathit{proj}(X)\}$

This approach would effectively remove the issue of continuity of $\bar{\mathrm{X}}$ projections from the realm of $\bar{\mathrm{X}}$ Theory. However, that is not quite so unintuitive as it might seem. In a sense $\bar{\mathrm{X}}$ Theory provides phrases with heads and maximal elements, and allows for the creation of any number of nodes in between them. However, it does not create those nodes except on demand from other modules. It is only from other modules that phrase-internal structure is motivated. The task of $\bar{\mathrm{X}}$ Theory is thus limited to assuring the agreement of features among members of a projection according to the Endocentricity Condition, and to defining the head and maximal projection.

The Continuity Condition (if we retain it as part of $\bar{\mathrm{X}}$ Theory) deals only with local neighbors, but it is incomplete and upwards-expanding, in that a node is expanded only into its mother, with no consideration of its sisters. The $\mathrm{X}^{\max}$ Condition and the Endocentricity Condition are the first examples we have seen of non-local configurations specified by a licensing condition. Each of these relates nodes somewhere in the projection to the head of the projection. Any number of nodes can intervene between the head and the node subject to these conditions. The $\mathrm{X}^{\max}$ Condition, in particular, departs from the kind of configurations we have seen in context-free licensing grammars, since it is non-local, incomplete, and upwards-directed.

5.1.3 Theta Theory Revisited

Current versions of Theta Theory (e.g., [Chomsky 1986b]) also involve the kind of discontinuous configurations which arise in the above version of $\bar{\mathrm{X}}$ Theory. In particular the definition of sisterhood is much relaxed, so as to allow all argument-slots to be discharged by the lexical head. The quasi-Categorial Theta Projection Condition offered above is in fact fairly non-classical in GB terms. However, in order to assign all the roles from one position it is necessary to be able to name an arbitrary role, something which we cannot do with the *arg1*-and-successor implementation of the grid used above. Another possibility involves adding to $\mathcal{L}$ some two-place function symbols, in particular the two-place function-symbol *arg*, which takes as its first argument a number, written in successor notation (e.g., 0, $s(0)$, $s(s(0))$, etc.), and as its second the predicate whose arguments are at issue. Thus, instead of writing

$$\mathit{arg1}(X), s(\mathit{arg1}(X)), s(s(\mathit{arg1}(X))),$$

we could write

$$\mathit{arg}(0, X), \mathit{arg}(s(0), X), \mathit{arg}(s(s(0)), X).$$

In effect the first argument of *arg* serves as a subscript. We can then refer to an arbitrary argument as $\mathit{arg}(N, X)$, for example, with a variable in the first position.

Accordingly, we replace the Sisterhood Condition and the Grid Projection Condition with the following.

(5.20) *Revised Sisterhood Condition*
$\mathit{arg}(N, X) \approx Y \Rightarrow$
$\{\mathit{i\text{-}dom}(V1, Y), \mathit{proj}(V1) \approx \mathit{proj}(X),$
$\mathit{dom}(\mathit{max}(\mathit{proj}(X)), V1), \mathit{dom}(V1, X)\}$

The Revised Sisterhood Condition simply states that if Y is some argument of X, it must be attached into the projection of X (under $V1$), below the maximal projection and above the head, that is, under what has traditionally been referred to as X'.

The condition in (20) allows for the kind of binary branching structure created by our earlier version of Theta Theory. However, it also allows all of X's arguments to be attached under the same node, creating an n-ary branching structure, where X has n arguments. We may wish to impose an asymmetrical structural hierarchy on arguments. To do so let us make our usual simplifying assumption that the argument-number corresponds to the associated argument's place in the hierarchy, so that $\mathit{arg}(0, X)$ is the most internal argument.

(5.21) *Thematic Hierarchy Condition*
$arg(s(N), X) \approx Y \Rightarrow$
$\{i\text{-}dom(V1, Y), i\text{-}dom(V2, arg(N, X)), dom(V1, V2)\}$

In this licensing condition, θ-role assignment is licensed as long as the higher numbered argument i-commands the next lower numbered argument (and, by transitivity of dominance, all lower numbered arguments).

Note that the Thematic Hierarchy Condition in (21) allows the domain in which the head's grid is discharged to be broken up by non-argument phrases, since it requires only that $V1$ dominate $V2$. This would allow adverbial phrases to intervene in a verb phrase between its objects. To rule this out it is only necessary to change the last formula of the environment to $i\text{-}dom(V1, V2)$. This will ensure that the θ-domain of the the head is continuous. This would, however, still allow adverbials to appear *below* all of a verb's arguments. If even this is to be ruled out, then a further condition is required.

(5.22) $arg(0, X) \approx Y \Rightarrow$
$\{i\text{-}dom(V1, Y), i\text{-}dom(V1, X)\}$

This condition simply stipulates that the node immediately dominating the head must also immediately dominate the lowest-ranked argument.

The Revised Sisterhood Condition of (20) is a very typical example of the kind of configuration specified by GB principles. It specifies a local structural relation of the argument to a node ($V1$) at which it is attached into its predicate's projection, but $V1$ is left to 'float' within its projection. Though this condition does supply clear and unequivocal structural information it also leaves a great deal unspecified. This gives a good indication of the motivation behind the use of logical descriptions in GB parsing research.

Note, however, that there is a problem to this approach to Theta Theory, which appears once we consider the new form of the Theta Criterion.

(5.23) *Revised Theta Criterion*
$\{cat(core(X)) \approx noun, X \approx max(proj(X))\} \Rightarrow$
$\{node(V1), X \approx arg(N, V1)\}$

Consider now what would happen if a licensing parser did not have available a binding for $V1$. Then, following the procedure we have assumed so far, it would create a new node, and substitute it for $V1$. So far, so good. However, there is another variable in the environment, which will certainly not be instantiated if $V1$ is not. I am referring, of course, to N, the argument number. Until now there has never been an occasion to attempt the arbitrary instantiation of variables standing for anything but nodes. The most

satisfactory response to this dilemma would be to find some way to prevent the parser from instantiating N, allowing it to remain a variable, either in the description, once the environment was asserted there, or else as part of a 'frozen daemon', a processing task which will suspend itself until all of its variables become bound. Clearly the descriptive leverage afforded by the use of variable argument positions is a very desirable thing, and extending the licensing parser model in this direction is a very exciting direction for future research. However, it should be noted that this approach is problematical for the parsing system presented in Chapter 3.

5.2 Case Theory and A-Chains

5.2.1 Case Theory

The theory of abstract Case (as opposed to morphological case) is concerned exclusively with the distribution of NPs (or DPs in many modern variants of the theory). The central principle of this Case Theory module is the Case Filter, which states that any NP with phonological content must receive Case. The module is fleshed out, in a manner somewhat analogous to Theta Theory, with a system of conditions on Case-assignment which regulate just where an NP can receive Case.

As a source of structure, Case Theory is much less satisfactory than Theta Theory. In most versions of GB Theory there are no less than three separate mechanisms of Case assignment. I will ignore inherent Case here, but even so this leaves us with two completely disjoint configurations in which Case can be assigned (under Government and under Spec-Head Agreement).

Nominative Case is assigned in conjunction with specifier-head agreement with finite inflection. That is, the Case-assignee is in the specifier position of the projection of its assigner, and aside from receiving Case, it also confers on its assigner its own agreement features.[2] Genitive Case (which I will not treat here in any detail) is also held to be assigned under a spec-head relation between the possessor NP and the determiner *'s*. Accusative Case, on the other hand, is assigned under government by the Case-assigner. Roughly, a head usually governs its complements, and the specifiers and heads of its complements (but not, on the version of GB adopted here, its own specifier).

This two-way disjunction, between Government and Spec-Head configurations, is made even worse by disjunctions inherent in the definition of

[2] Whether agreement is a form of feature assignment or feature matching is irrelevant here. The equational language in which feature identifications are made in $\mathcal{L}$ is completely neutral between these two possibilities.

government. Thus a head can govern its arguments, the specifiers of its arguments, and the heads of its arguments. Even though this last possibility is not relevant for Case-marking (it matters for Head-Movement, which we will not investigate here), this still leaves a rather disparate collection of governable positions. There seems to be no succinct way of defining these positions as a natural class in $\mathcal{L}$ as currently implemented (though see below, p. 121). Therefore it would appear that a licensing parser of the kind being investigated here will have to explore a large number of options as it seeks to impose a Case-marking configuration on the surroundings of a lexical NP.

Because of the large number of possible Case-marked positions around a head, the uniqueness of Case-assignment becomes a real issue. It is not permitted for an argument to be assigned more than one Case, any more than it is permitted to receive more than one θ-role, yet it is very easy to imagine positions where a single NP could receive Case from two assigners. Take for example the specifier position, where an NP could receive Case under Agreement from the head of the projection of which it was the specifier, and Case under Government from a higher head, whose complement was the projection including the NP. The possibility of multiple Cases becomes greater when we consider the possibility of Movement, or A-Chain Formation. Consider the difference in acceptability between (24) and (25).

(5.24) *John$_i$ seems t$_i$ to have succeeded.*

(5.25) **John$_i$ seems t$_i$ has succeeded.*

In (25), the finite inflection of *has* assigns Case to *John* (or, more accurately, to the trace t$_i$ left by *John*). Therefore *John* gets Case from (the finite inflection on) both *seems* and *has*.

Note that if one Case-assigner assigned, say, accusative, and another assigned, say, nominative to an NP, then no problem would arise: the resulting description would support the conclusion that, since the Case of NP equals both nominative and accusative, then nominative equals accusative, which leads directly to a contradiction. However, as we see in example (25), it is possible for multiple Case-assigners to assign the same Case (nominative, in this example). Therefore some more powerful mechanism seems necessary. Nonetheless, the basic strategy of appealing to a feature-clash is one which we can still use here. Suppose that every Case-assigner has associated with it a unique index. We thus make the *case* feature complex; it now has both a *label* (e.g., nominative, dative, etc.), and an *index*.

(5.26)
$$\begin{aligned} label(case(\alpha)) &\approx acc \\ index(case(\alpha)) &\approx 73 \end{aligned}$$

The result of assigning Case from α to β will be that β now has the function $index \circ case$ defined for it. If some other γ assigns Case to β, then $index(case(\beta))$ will be multiply defined, leading to a contradiction of the familiar sort. Note that the same device can be used to assure the uniqueness of θ-roles, where here each *grid position* is assigned a unique index.

Structural Case-marking properties are independent of a node's argument structure grid (as opposed to inherent Case-marking properties, which are tied to grid positions). Hence I will write them as a function of the node itself, analogous to the *grid* and *core* features.

In addition to the type of Case being assigned, and the index of the Case-assigner, some indication is needed as to whether an element is a Case-assigner at all. So, for instance, a passive participle is not able to assign structural Case. Nor are intransitive verbs (in most instances), or verbs taking only prepositional or clausal arguments (such as *talk*, or *hope*). Furthermore, the type of Case-assignment possible for any given assigner is strictly circumscribed. Thus, in English, finite inflection can only assign Case under Spec-Head Agreement and verbs and prepositions capable of assigning structural Case can only assign it under Government. The ability to assign Case thus appears to be a lexical property. Hence it is to be marked as a feature of lexical elements, as follows. If an element is not a Case assigner, it will have the property that $case(\alpha) \approx nil$. If an element is a Case assigner, then we will specify for it the *index* and *label* features mentioned above in (26). If the application of a licensing condition to an NP then imposes case-features on a neighboring intransitive V, for example, this will result in the following system of equations.

(5.27)
$$\begin{aligned} &case(V) \approx nil \\ &index(case(V)) \approx \iota \quad \text{(an arbitrary index)} \\ &\vdash index(nil) \approx \iota \end{aligned}$$

As long as *nil* is an element of **Feature-Values**, then any function applied to it will yield $\perp$, by axiom $\mathcal{T}222$. Since $index(nil) = \perp$, it cannot $\approx$ anything, so the resulting description, in which a non-Case-assigner is forced to assign Case, is inconsistent.

The Case Filter states that all lexical NPs must be Case-marked. More accurately, all NP-Chains must be Case-marked, but we will not turn to the examination of chains until the next section. Accordingly, we want a licensing condition of a form like the following.

(5.28) *Case Filter*

$$\{cat(core(X)) \approx noun, X \approx max(proj(X)), lexical(X) \approx +\}$$
$$\Rightarrow$$
$$\{node(Y), Y \approx head(proj(Y)),$$
$$index(case(Y)) \approx index(case(Y)),$$
$$case(core(X)) \approx case(Y)\}$$

The first two lines of the environment specify that the Case-assigner Y must be a lexical-level node which is a Case-assigner. The rather trivial looking reflexive equation in the second line is actually not so trivial in light of the discussion above, since it forces the feature *index* to be definable for $case(Y)$. Therefore this description is not consistent with the description of a non-Case-assigner. Note that we cannot use the somewhat more compact $\alpha \approx \_$ notation for 'is defined', since it would introduce an arbitrary variable into the environment which does not stand for a node. (Recall the discussion of the Revised Theta Theory, above.)

We then require an auxiliary condition which characterizes the structural environments where an NP can get Case. The environment of this condition divides up into two disjunctions, according to the type of the Case assigner. Note that, in English, the type of Case assignment correlates with the status of the Case-assigner as a functional or lexical category. Accordingly we add to $\mathcal{L}$ the function symbol *func*, which takes the categories of nodes to either $+$ or $-$. This means that we cannot refer to noun phrases with $cat(X) \approx noun$, since we cannot define any function which might apply to a feature value. Instead we would need to adopt the standard feature system, using $[\pm N, \pm V]$. That is, a nominal category (N or NP, e.g.) would be characterized as follows.

$$\begin{aligned} noun(cat(core(X))) &\approx + \\ verb(cat(core(X))) &\approx - \\ func(cat(core(X))) &\approx - \end{aligned}$$

Further, the sub-condition controlling Case assignment under Government divides up into two more disjuncts, according to whether Case is assigned by a head to its complement or to the specifier of its complement.

(5.29) *Structural Conditions on Case Assignment*

$$case(core(X)) \approx case(Y) \Rightarrow$$
$$\{func(cat(core(Y))) \approx -, X \approx arg1(Y), i\text{-}prec(Y, X)\}$$
$$\vee$$
$$\{func(cat(core(Y))) \approx -, X \approx spec(proj(arg1(Y))), i\text{-}prec(Y, X)\}$$
$$\vee$$
$$\{func(cat(core(Y))) \approx +, X \approx spec(proj(Y)), phi(core(X)) \approx phi(core(Y)), i\text{-}prec(X, Y)\}$$

In this implementation, government is restricted to i-commanded positions, since only the first argument of Y is mentioned by the relevant sub-conditions. This condition can be relaxed by introducing a new variable $V1$, which is a projection of the head Y (possibly identical to $V1$). Then government could be characterized as holding between Y and $arg1(V1)$ (or its specifier). In the third sub-condition, dealing with Spec-Head Agreement, the features shared are the *phi*-features, which, properly speaking, include some features like gender that are not overtly marked in English agreement. Note that the identification cannot be made any higher up in the feature structure, since category features are dependents of *core*, and NP and Infl are, obviously, categorially distinct.

The last line of each sub-condition implements Stowell's Adjacency Condition on Case Assignment [Stowell 1981]. Ideally this would be expressed as a separate condition. However, the choice between $i\text{-}prec(Y, X)$ and $i\text{-}prec(X, Y)$ is so directly tied to type of Case-assignment that it makes sense to place it directly into the other structural conditions on Case assignment. Note that if we express it separately, we would have a condition like the following.

(5.30) *Adjacency Condition*

$$case(core(X)) \approx case(Y) \Rightarrow$$
$$i\text{-}prec(X, Y)$$
$$\vee$$
$$i\text{-}prec(Y, X)$$

Thus, for each of the three types of Case assignment there would be two additional possibilities, one of which should never be allowed. Therefore it seems appropriate to implement this condition directly as part of the other structural conditions on Case.

There is a way to represent the governable positions more succinctly, but it gets us into the same problems as the two-place *arg* function discussed above. Recall that government essentially applies to three distinct positions: the maximal projection, the specifier of that projection, and the head of that projection.[3] This last possibility is only relevant for head-movement, which I will not investigate here, but including it in this discussion does not add to our difficulties. The important thing to note is that these three elements, maximal projection, specifier and head, are exactly the three uniquely specifiable elements of a projection, as we have defined it above. We could thus make the generalization, considering a feature structure as a sort of graph, that the government domain is essentially the sub-graph rooted in $\mathit{proj}(X)$. If we could only write terms like

$$F(\mathit{proj}(X)),$$

where we take F to be a variable ranging over function-names, then we could use such a term to express exactly those positions which can be L-marked by a head. Unfortunately, allowing ourselves to use variables ranging over functions draws us out of the bounds of first-order logic. There is a trick which will allow us to get around this restriction; it is the same one we used above in our revisions of the Sisterhood configuration, and, unfortunately, makes the same as-yet-unsatisfied demands on the parsing model. If we allow two-place function names into $\mathcal{L}$, then we can treat the symbols *max*, *head*, and *spec* as naming constants which combine with projections to pick out the relevant nodes. I will refer to these as "$\bar{\mathrm{X}}$ grammatical functions", and employ the new, two-place function name *gf* for these purposes. Accordingly we would have terms like the following.

$$\mathit{gf}(\mathit{max}, \mathit{proj}(X)), \mathit{gf}(\mathit{head}, \mathit{proj}(X)), \mathit{gf}(\mathit{spec}, \mathit{proj}(X))$$

The configuration relevant to government can then be concisely expressed as:

$$\mathit{gf}(\_, \mathit{proj}(X)).$$

Accordingly we have, as the part of the licensing environment dealing with government, the following.

(5.31) $\mathit{case}(\mathit{core}(X)) \approx \mathit{case}(Y) \Rightarrow$
$\{\mathit{func}(\mathit{cat}(\mathit{core}(Y))) \approx -,$
$X \approx \mathit{gf}(F, \mathit{proj}(\mathit{arg}(0, Y))),$
$\mathit{i\text{-}prec}(Y, X)\}$

[3] In fact, in a full treatment of government, in a system allowing adjunction to maximal projections, for example, it would not be *government* itself that we are characterizing, but rather *L-marking*, which is a subsidiary concept. However, it is still the main source of disjunctions, so the potential savings should carry over.

Once again, the restriction of structural Case assignment to direct complements of heads has roughly the effect of the i-command condition on government in [Sportiche 1990]. If we wished to allow structural Case assignment to any complement, we would replace $\mathit{arg}(0, Y)$ with $\mathit{arg}(\_, Y)$, but this does not seem restrictive enough to cover the facts of structural Case assignment.

Unfortunately, as with the revised Theta Criterion, this revision of the conditions on Case assignment will only work with our current parsing model if we can guarantee that the variable F will be bound, which in turn means that the governee, X, must already have a fixed position within its containing projection, and this projection must already be attached as the direct complement of the Case-assigner, Y, before this condition can apply. Clearly, though, the extra expressive power gained by the use of the two-place *arg* and *gf* functions makes the appropriate modifications an exciting possibility for future research.

5.2.2 Chain Theory and Trace Theory

It is perfectly possible, given the theory so far, that Case Theory and Theta Theory can make conflicting demands of the NPs which come under their sway. This is the basis of the GB analysis of passive and raising constructions, for example, in which Theta Theory requires an NP in object position, but Case Theory requires it in subject position. Languages solve the problem of these conflicting constraints by, in effect, stretching the NP so that it can occupy both positions. There are two very closely related approaches to making NPs stretch in this way. The most common approach says that the NP occupies these positions at different levels of representation; it satisfies Theta Theory at d-structure, and Case Theory at s-structure. An alternative formulation would have the NP satisfy these conditions through Chain Formation, so that the Chain satisfies the conditions, rather than the NP itself. I will pursue this latter version here, more because it allows us to restrict our attention to a single level of representation (s-structure) than because there are any deep issues of principle involved.

A chain, like a projection, is an organization of nodes. Thus, in the same way that, for a node X, we can speak of $\mathit{proj}(X)$, so also we can speak of $\mathit{chain}(X)$. In addition, like projections, chains have various distinguished elements. In particular we are interested in the *foot* of the chain, which is that element of the chain which is structurally lowest, that is, which is c-commanded by all the others, and, analogously, we are interested in the *head* of the chain, the highest element, the one which c-commands all the others.

Various conditions hold of well-formed chains. First, at most one element of the chain can be lexical. As long as our little grammar sanctions only A-chains, then we can say exactly one element of the chain can be lexical. We capture this two way implication in two licensing conditions, one licensing [+lexical] elements by making them heads of chains, and one licensing heads of chains by insisting they be [+lexical]. Note that for perfect harmony with the other conditions of this chapter we should restrict these conditions to NPs, however, for simplicity, I will ignore this extra restriction here.

(5.32) Visibility Condition
$lexical(X) \approx +$
$\Rightarrow$
$X \approx head(chain(V1))$

(5.33) Condition on Chain Heads
$X \approx head(chain(Y))$
$\Rightarrow$
$lexical(X) \approx +$

I have referred to condition (32) as the *Visibility Condition* for the following reasons. The notion of a Visibility Condition arises originally from a suggestion of Aoun [Aoun 1979], and is incorporated into [Chomsky 1981]. This condition was originally seen as a way to derive the Case Filter as a condition on the proper thematic interpretation of NPs (here chains): the chain is not 'visible for θ-role assignment' unless it is Case-marked. Adapting somewhat freely to present circumstances, it is the lexicality of NPs which both subjects them to the Case Filter, and also triggers their absorption into a chain. Note that the Theta Criterion (7), which requires NPs to be θ-marked, must now be revised since NPs may not be in θ-positions at s-structure. Let us assume that lexical NPs are assigned Case as heads of chains, and θ-roles are assigned to chain feet. This will be so even when the chain head and foot are later identified with each other, in cases where no movement has actually taken place. This greatly simplifies trace-insertion, but it has as a side-effect the following consequence: the only way for lexical elements to receive a θ-role is through chain-formation, even in constructions where no movement has taken place. (I will discuss this latter point in more detail below.) If we take the property of being [+lexical] to correspond roughly to the concept of 'having reference,' then chain formation, even if vacuous, is necessary to associate a role and its referent. Hence what I have called the Visibility Condition here really does do something very much like connecting Case and θ-role assignment, the task which underlies the classical Visibility Condition.

As just mentioned, we must amend the Theta Criterion, since not all NPs will in fact be in a sisterhood relation to their θ-assigners, in particular the heads of chains sometimes will not. The Theta Criterion thus becomes a condition not on NPs (or argument-type phrases) but on *chain feet.*

(5.34) *Revised Theta Criterion*
$\{X \approx head(chain(X)),$
$verb(cat(core(X))) \approx -,$
$noun(cat(core(X))) \approx +\}$
$\Rightarrow$
$\{node(V1), foot(chain(X)) \approx arg1(V1)\}$

So far nothing prevents chains where the head is identical to the foot. Indeed, given the revised Theta Criterion, it must be the case that some NPs (those that don't move) are in chains whose head and foot are the same. We are essentially treating all NPs as chains. The cases where the head equals the foot are essentially just special cases. This is the tack that [Chomsky 1988] takes in characterizing the level of LF, for example. Note that to do this we will need a theory of chains powerful enough to make this identification of head and foot when it is warranted. Thus we need the following condition on chains, which states roughly that if an element is a member of a chain, then that chain is either simple or complex. That is, NPs, which are subject to both the Theta Criterion and the Case Filter can be licensed by either the identification of their chain's head and foot, or by (true) chain-formation. A chain which is complex has a foot which is a *trace.* We here denote traces as being explicitly the trace of something in particular, using our familiar functional notation.

(5.35) *Trace Insertion Condition*
$chain(X) \approx \_$
$\Rightarrow$
$\{head(chain(X)) \approx foot(chain(X))\}$
$\vee$
$\{node(V1), foot(chain(X)) \approx trace(V1)\}$

All the non-lexical elements of chains are traces. These are essentially dummy copies of the lexical element of the chain, which do not dominate any finer-grained structure. This makes them slightly anomalous with respect to $\bar{X}$ Theory, as we might expect, since $\bar{X}$ Theory is supposed to apply at d-structure. Nonetheless, since we are interested in a simplified theory which has only a single level, we are forced to make allowances in $\bar{X}$ Theory to allow for the existence of traces.

If the moved element (the head of the chain) is a maximal projection, then its traces must be also. However, since they dominate no structure, they are maximal elements of a projection with no other members. Looking back at our version of $\bar{\text{X}}$ Theory, note that the Continuity Condition requires only that a node either be the maximal projection or be dominated by another projection. As long as traces are maximal projections they satisfy this condition. They do not trigger the X^{max} Projection Condition, since they are not lexical (and they are not heads in any case). As a result a trace category should emerge from the application of $\bar{\text{X}}$ Theory with no specification for the feature *lexical*. In particular they will not be marked $lexical(X) \approx +$.

Note, however, that the Endocentricity Condition as currently written does cause problems in a representation with chains. If $max(proj(X))$ is defined, then we have in addition that $proj(X) \approx proj(X)$, by axiom $\mathcal{T}274$. Therefore traces will trigger the Endocentricity Condition, with the side-effect that $head(proj(X))$ will be defined (as the source of core features for all the elements in a projection). Traces differ from most $\bar{\text{X}}$ projections in that they get their core features from their antecedent, not from a lexical head. One possible solution is to require that there already be more than one element in the projection before the Endocentricity Condition can apply.

(5.36) Revised Endocentricity Condition

$$proj(X) \approx proj(Y)$$
$$\Rightarrow$$
$$core(X) \approx core(Y)$$

The revised condition requires either that the projection contain more than one node, or that X equal Y, which is a harmless case, since it just causes X's core features to be asserted to be self-identical.

The one lexical element of a chain must be the head of the chain, and, since it is lexical, it must be Case marked. So the head of a chain will always be a Case-marked position. Since we regard a chain as an extension of an NP, we insist that, just as an NP can only have a single Case and a single θ-role, so also a chain's Case and θ-role must each be unique to that chain. Since Case must be assigned to the head, and a θ-role must be assigned to the foot, it follows that all other elements of the chain must appear in positions which are neither Case- nor θ-marked. One way to guarantee the uniqueness of a chain's Case and θ-role is to have the chain inherit these features directly from all of its members. That is, we define the *core* function to apply to chains as well as to nodes (since chains are after all a sort of extension of a node).

(5.37) *Chain Agreement Principle*

$$chain(X) \approx \_\_$$
$$\Rightarrow$$
$$core(chain(X)) \approx core(X)$$

The Chain Agreement Principle is stronger than we need to guarantee that θ-role and Case will be unique. However, the extra strength is warranted, since we want the elements of a chain to share other features as well. Obviously we want them to share category features. We may also want them to share ϕ-features, so as to control agreement with, e.g., reflexive pronouns which are local to a trace but not to its antecedent.

Two more conditions govern the shape of chains, but as licensing conditions they are better viewed as conditions on traces. First, traces must be bound by a c-commanding antecedent, and, second, they must be bound within a local domain. These two configurations are commonly combined into an Antecedent Government Condition on traces. Note that here 'government' is not quite the same configuration as in Case assignment, where government by a head was at issue. Here it is government of one maximal projection by another which is at issue. Nonetheless the two configurations are related, in that, in general, an NP antecedent will govern a trace if the head of the projection in which the antecedent is attached governs the trace, in the sense already discussed.[4]

The positions in which the antecedents of traces can appear are strictly circumscribed by the theory of phrase structure underlying this simple sample grammar we are investigating here. Since the antecedent of a trace cannot be in a θ-marked position, it cannot appear as an argument of a head. In our system here this leaves only specifier positions as possible targets of movement. (A fuller treatment of XP movement would allow also for adjunction to X^{max}, but this would require an extremely lengthy treatment, and is beyond the scope of this dissertation.) Therefore, for present purposes, we can consider Antecedent Government to be government out of specifier positions. Except for the choice of governor, however, the details of this configuration are essentially the same as those for Head Government in Case Theory. Using the more expressive two-place notation, we can state the condition quite succinctly.

(5.38) Antecedent Government Condition (1)

$$X \approx trace(Y)$$
$$\Rightarrow$$
$$\{Y \approx gf(spec, proj(V1)),$$
$$X \approx gf(\_\_, proj(arg(\_\_, V1)))\}$$

Without the extra expressive power of this '$\mathcal{L}'$', we need to spell out the

[4] This is not quite true if we maintain an i-command condition on government, and adopt a binary branching structure, with asymmetric c-command relations among arguments, since then the XP in Spec position can govern the outer arguments of the head, and the head can only govern its innermost argument.

disjunctive possibilities in detail, giving us the condition in (39).

(5.39) Antecedent Government Condition

$$X \approx \mathit{trace}(Y)$$
$$\Rightarrow$$
$$\{\mathit{node}(V1), Y \approx \mathit{spec}(\mathit{proj}(V1)),$$
$$X \approx \mathit{arg1}(V1)\}$$
$$\vee$$
$$\{\mathit{node}(V1), Y \approx \mathit{spec}(\mathit{proj}(V1)),$$
$$X \approx \mathit{spec}(\mathit{proj}(\mathit{arg1}(V1)))\}$$

This concludes our investigation of GB Theory through the lens of the Licensing Grammar formalism and parsing model. It is clear at this point that, though the formalism is clearly too powerful to express phrase-structure grammars with succinctness and elegance, it is nonetheless not quite powerful enough to capture some of the most characteristic configurations of GB Theory. For example, the government configuration (the 'G' of 'GB'), though it can be expressed, essentially by enumerating the possible targets of government, cannot be expressed as compactly as we would like. Nonetheless, it is fairly clear along what lines the parsing model would have to be improved, to support the kinds of licensing conditions we would really like.

There are, however, other aspects of the theory which should prove even harder to accommodate in a licensing parser. First, recall that nothing has been said here about the representation of XP adjunction, as it arises in Ā Movement. In particular, phrases which move by adjoining to another phrase are potentially governed in much the same way that specifiers are. However, there is nothing about adjunction which would make adjuncts unique in the way that specifiers are, in which case we could never guarantee that a term like $\mathit{gf}(\mathit{adjunct}, \mathit{proj}(X))$ was well-defined. Recall, however, that what I have called 'government' in this text is in fact much closer to the concept of *L-marking* or *L-dependence* in fuller treatments of government [Chomsky 1986b, Sportiche 1990]. The definition of government then depends on the definition of 'barrier', which then depends on the definition of L-marking. Clearly I will have to follow the theory in this matter, and capture these more complex cases with extra layers of conditioning.

Another problem involves Binding Theory. Though the details are not clear, it seems probable that some account of locality conditions along the lines of Principle A could be given in a logical description. It seems also possible in principle to do the same for Principle B, along the following lines: given a pronoun, assume as its licensing environment a description of the domain in which it must be free, and add to that its antecedent, with the proviso that it be excluded by that domain. However, note that this makes

crucial reference to the antecedent. If a pronoun is free in the whole sentence, then this kind of licensing condition could not work. Only a system of logical descriptions which allowed prohibitions, in addition to purely constructive, positive assertions, could express such a configuration.

Another Binding-related problem involves c-command. It is quite easy to express i-command using logical descriptions, and even m-command is not so difficult, but c-command is completely impossible given the current definition of logical description. The problem is the crucial role played in the definition of c-command by the first branching node. Marking the first, that is, lowest in the dominance order, element satisfying an arbitrary property (e.g., that it branch) can only be done by allowing quantifiers into descriptions. Note that if we did not subject the *i-dom* predicate to quantified consistency conditions (axiom $\mathcal{T}287$), we could not express even immediate dominance in the parts of $\mathcal{L}$ remaining to us. The problem represented by c-command is just a generalization of the problem of characterizing 'the minimal (or maximal) element N such that $P(N)$,' for an arbitrary property P.[5]

The analogy with immediate dominance and precedence suggests the lines along which logical descriptions must be extended to allow for these more powerful descriptions. Immediate dominance, by virtue of the presence of $\mathcal{T}28(7)$i among the conditions on description consistency, expresses certain notions which are inherently quantificational. This is accomplished in this preliminary version of $\mathcal{L}$ by pairing a simple atomic formula like $\mathit{i\text{-}dom}(\alpha, \beta)$ with a complex statement in the full first-order description logic (i.e., $\mathcal{T}28(7)$i). In order to allow more occurrences of inherently quantified expressions into logical descriptions we need a more general form of this same approach, so that certain formulas in the description are essentially stand-ins for more complex conditions, which are treated essentially as extensions of the consistency checker.

One possibility might be to introduce generalized variable-binding operators of a form like

$$\mathit{min}(X, \psi[X]),$$

where $\psi[X]$ is a logical description in a quantifier-free version of $\mathcal{L}$ in which the variable X appears. This could be tied to a consistency condition like

$$\mathit{min}(X, \psi[X]) \leftrightarrow \psi[X] \wedge (\forall Y \psi[Y] \rightarrow \mathit{dom}(Y, X)).$$

Given such a predicate (properly, a *schema* for a collection of predicates, which are created by the grammar-author as needed), we could recast the special case of $\mathit{i\text{-}dom}(X, Y)$ as $\mathit{min}(X, \mathit{dom}(X, Y))$. I am not certain whether

[5] Thus immediate dominance is the special case where $P = \mathit{dom}$.

this specific dodge would work (certainly it would introduce a severe computational burden in parsing). However, it seems clear that this basic scheme, of treating quantified descriptions as essentially daemons monitoring the consistency of the parser's working description with a simple, atomic logical description of the familiar sort (ψ, in this example), is the most likely line of least resistance in increasing the expressive power of Licensing Grammars and Licensing Parsers.

Though there are aspects of GB Theory which clearly lie outside the expressive power of the description and licensing theories advanced in this dissertation, it should be clear that even to capture the parts of the theory which we have been able to here it has been necessary to drastically relax the conditions on licensing conditions which characterize rewrite grammars. Expansion is not always downwards in these examples—in fact it is more often upwards—nor is it by any means complete or unique, or strictly local. It is a strong claim of GB Theory that the crucial, deep properties of syntactic structures cannot be grasped without lifting up our gaze from the narrow neighborhoods envisioned in immediate constituent analysis. The underlying principles of syntax can only be characterized in terms of configurations which are overlapping, and which can have gaps in them. It is my hope that through continuing work on Licensing Grammars, computational linguists can face these theoretical claims squarely, without thereby giving up the kind of constructive parsing schemes with which the field has had such success in dealing with simpler kinds of grammars.

Chapter 6

A Licensing Theoretic Account of Asyntactic Comprehension in Broca's Aphasics

In Chapter 3, I define well-formedness for licensing grammars. In this chapter I will investigate grammaticality judgment.[1] Usually we do not distinguish these two concepts, especially since most of the data which bears on the precise definition of well-formedness is drawn from grammaticality judgments of native speakers.[2] The normal assumption is that grammaticality judgment is essentially a process of determining whether the grammar assigns to the stimulus a well-formed, i.e., fully licensed syntactic structure. However, there is empirical data from the study of neurologically impaired populations that suggest a somewhat richer picture than this.

These data have long suggested to researchers that a certain population of aphasics, the agrammatic Broca's aphasics, are suffering an impairment which prevents them from achieving a fully detailed syntactic representation of input sentences. The data show a syntax-based comprehension deficit

[1] The work presented in this chapter owes a great deal to a long and fruitful collaboration with V.A. Fromkin and G. Mauner. Some of this discussion has appeared in [Cornell 1989, Cornell 1990, Mauner 1990]. The presentation here pretty much follows that of [Cornell 1992].

[2] It would be slightly more accurate to say "acceptability judgment," rather than "grammaticality judgment," since we do not have *a priori* knowledge that a subject's judgment is due to his or her grammar. The strongest argument we can advance that a subject's judgments are in fact grammaticality judgments is to offer an insightful theory of their grammars which has these judgments as consequences. Here I will be attempting such an analysis, according to which these subjects' judgments *do* follow from the application of their grammars to the parsing and judgment problems, so I will for the most part blithely refer to the relevant task as grammaticality judgment without further argument.

(termed *asyntactic comprehension*), which co-occurs with remarkably good grammaticality judgment ability on the very same structures which cause comprehension difficulty.

As a result of the work presented in earlier chapters (in particular Chs. 2 and 5), we are in a position to approach this very difficult problem with a very high degree of formal rigor. In particular the theory of partial descriptions of Ch. 2 will allow us to say how these subjects can judge partial descriptions which are missing information crucial to comprehension but inessential for judgment, appearing as a result to be performing normal-like syntactic processing of these constructions.

6.1 Agrammatism, and an Apparent Paradox

6.1.1 Agrammatic Speech and Asyntactic Comprehension

The term *agrammatism* properly applies to a speech production phenomenon commonest among Broca's aphasics. In general, the speech of such subjects is effortful and non-fluent. However the characteristic which in recent years has become definitional is the loss of the functional vocabulary of the grammar (e.g. determiners, auxiliaries, pronouns and bound inflection), with comparatively greater retention in free speech of the "open class", content words. Some samples of agrammatic speech, taken from [Linebarger 1983a], are given in (1), below. The examples in (2) are from [Schwartz 1985]. Each example represents the speech of a different subject. They are responses to the Cookie Theft picture of the Boston Diagnostic Aphasia Test [Goodglass 1972].[3] Note that in these examples the functional vocabulary is not completely lost, and that subjects vary to some degree in the severity of their deficits. The *-ing* form of the verb is quite common in almost all of the examples, but the progressive auxiliary appears more commonly in the speech of the first and third subjects than in the second. It should also be noted that agrammatics' production problems are not all limited to the production of closed class elements. In the examples in (2), subjects are generally attempting to produce some sort of dative construction. This is clearly very difficult for them, and the difficulty does not seem to hinge on the proper use of a functional element. Here the difficulty is even more

[3] This picture depicts a kitchen scene, in which a mother is washing dishes while, behind her back, her children are attempting a cookie-jar raid. The step-ladder is about to tip over, out from underneath the cookie thief, and meanwhile the sink is beginning to overflow, while the mother gazes out the window.

clearly syntactic in nature.

(6.1) *Cookie Theft*

- (a) "Mother is washing the ... dishes... The David and Kathy ... cookies. Fall on. The ... the ... water. The water is ... clump ... water is ... flowing. The chair is ... the ... the boy is counter ... no, fall off."
- (b) "A mother ... a dish ... drying..." (What?) "Plate ... a faucet ... running ... a boy, eating cookies ... eating the cookies ... girl."
- (c) "The boy is falling on the ... almost falling floor. The mother is running ... faucet ... floor. The sister is passing cookie." (What's the mother doing?) "Washing the things. Almost full, splashing water..."

(6.2) *Valentine's Day*

- (a) "The boy is gave ... The boy is gave the card."
- (b) "The boy show a Valentine's day ... The boy and the girl is valentine."
- (c) "The girl... the boy is giving a... giving his girlfriend. The boy valentine the girl. the boy givin' valentine to girl."
- (d) "The boy is valentine the girl. The boy is giving the valentine and girl pleased."
- (e) "Valentine's day and candy. I think Valentine's day. Girl is Valentine's day... Boy is getting with the girl valentine's candy."

Broca's aphasics were long felt to exhibit problems in production, but no comparable problems in comprehension. However, a number of recent experiments have shown that, where comprehension depends on syntactic structure, agrammatics often show a pronounced deficit.[4] In particular, one group of subjects showed a distinction in their comprehension abilities which admits of an especially simple linguistic description.[5] Thus, in picture-pointing tasks designed to elicit recognition of the thematic content of a sentence ("who verb-ed whom"), these subjects performed pretty much like normals when presented with active and subject-relative clauses, but performed at chance

[4] Cf. [Caramazza 1976, Heilman 1976, Schwartz 1980].
[5] Cf. [Grodzinsky 1984, Grodzinsky 1986, Grodzinsky 1990, Caplan 1986].

on passives and object-relatives. That is, roughly half the time they interpreted passive sentences like (3b) as if they were in the active voice, with the boy doing the chasing.

(6.3) *Asyntactic comprehension*

(a) Active: *The boy is chasing the girl.*

(b) Passive: *The boy is chased by the girl.*

(c) Subject Relative: *The boy that is chasing the girl is slow.*

(d) Object Relative: *The boy that the girl is chasing is slow.*

Descriptively, what seems to be happening is that subjects' performance drops steeply whenever the direct object has been extracted. When the direct object surfaces in its canonical post-verbal position, thematic role assignment proceeds to the correct result.[6] Taking this phenomenon of *asyntactic comprehension* together with the observation that agrammatics have trouble producing functional categories, a category defined in syntactic terms, the obvious conclusion to reach was that agrammatic aphasics suffer a specifically syntactic deficit.

6.1.2 Agrammatics' Grammaticality Judgments

Pursuing this hypothesis, Linebarger et al. used a grammaticality judgment task in their experiments.[7] It is generally felt in the linguistic community that the grammaticality judgment task offers fairly direct evidence of people's linguistic (in particular syntactic) knowledge. If that knowledge base were disrupted, or if the routines which put it in play were damaged, then the judgment task ought to point it out clearly. A few examples of the kinds of stimuli which subjects were required to judge are given in (4). It should be clear that only syntactic knowledge should avail to produce the correct responses—a grasp solely of the semantics of the individual words should be insufficient.

(6.4) *Grammaticality judgments*

(a) *He came my house at six o'clock.

(b) I want/*hope you to go to the store now.

[6] This is not quite the same thing as thematic role assignment proceeding normally. Just because they achieve the correct result does not mean that the result is achieved by normal means, cf. [Cornell 1990, Caplan 1986].

[7] [Linebarger 1983a, Linebarger 1989, Linebarger 1990]

(c) I hope/*want you will go to the store now.

(d) *She went the stairs up in a hurry.

(e) She rolled the carpet up in a hurry.

(f) *Did the old man enjoying the view?

(g) *This job was expected Frank to get.

(h) *The workmen were expected would finish by noon.

(i) *Who did you think would invite?

(j) *Mary ate the bread that I baked a cake.

(k) *The paper was full mistakes.

Surprisingly, subjects appeared to perform extremely well on these tasks. According to Linebarger et al.'s measurements of subjects' sensitivities, they performed almost like normals on almost all conditions, and certainly well above the chance level of performance which they exhibited in comprehension tasks.[8] Thus we are presented with an apparent paradox. Comprehension performance indicates, at least for some subjects, that they have a problem in executing a correct, normal-like parse of the input sentence. Grammaticality judgment performance, however, indicates that they have little trouble in determining the structural, syntactic well-formedness of sentences. They could not do this if they could not build the syntactic structures of those sentences.

It gets worse. If we consider examples (44h) and (44i), we note that subjects must be sensitive to many of the conditions on chain formation. In (44h), the chain headed by *the workmen* gets Case twice (from *were* and *would*). In (44i), the chain headed by *who* binds only one of the empty elements in the lower clause, and the other violates the condition on Antecedent Government, since it has no antecedent.[9] To be sensitive to these violations, subjects must be able to construct chains, or, to put it another way, they must be sensitive to the conditions which govern chain-formation.

[8] Note that the subjects which Linebarger et al. were studying included some subjects who showed the pattern of asyntactic comprehension discussed above, but also some who showed difficulty comprehending active voice sentences as well as passive voice sentences. It is generally assumed that this latter group is suffering from a more severe form of the same deficit, even though the overt evidence for a syntax-based dissociation is lacking in their performance.

[9] Theoretically, sentence (44i) could just as easily be treated as a Projection Principle violation. However, given the formulations of Theta Theory in Chapter 5, it should in fact be the Antecedent Government condition (38) which is violated, since the Projection Principle's effect is just to posit a node, leaving it up to other conditions to determine if that node is a trace or a lexical element. Since it is not a lexical element, it will be treated as a trace.

Subjects show similarly preserved abilities in judging passives.[10] That is, they are able to discriminate the relative acceptability of sentences like the following.

(6.5) *Judging Passive Morphology*

(a) John has finally kissed Louise.

(b) *John was finally kissed Louise.

(c) The boy was followed by the girl.

(d) *The boy was followed the girl.

(e) The boy was following the girl.

Subjects must be sensitive to the Case-marking properties of passive participles to rule out examples (55b) and (55d). Further, they must be able to deduce that *kissed* and *followed* are passive participles in the relevant examples, so they must be able to tell which auxiliary was used, they must recognize the ending on the participle, and they must be able to access enough of their grammatical knowledge to deduce from these data that the clause is in the passive voice. Therefore, though they exhibit difficulty in *producing* closed class elements, it is difficult to see that they have an equivalent difficulty in *recognizing* them.

However, Linebarger and her colleagues did discover some constructions which *were* difficult to judge. Subjects had pronounced difficulty in judging tag-questions, as in (6), below, either when the subject of the tag disagreed with the initial subject, or when the auxiliary of the tag disagreed with the initial auxiliary.

(6.6) *Tag Questions*

(a) The blonde woman is outside, isn't she?

(b) *The blonde woman is outside, isn't it?

(c) John is very tall, isn't he?

(d) John is very tall, doesn't he?

Subjects show the same difficulties with agreement (either gender and number agreement in pronouns, or 'auxiliary-type' agreement in modals and

Leaving details of formalization aside, the nature of the deformity is clearly the failure of one of the arguments of *invite* to be properly *identified*, either via a supporting string of lexical items, or by an antecedent.

[10] Cf. [Linebarger 1990].

auxiliaries) in other constructions. In (7), below, subjects show difficulty enforcing agreement of reflexives with their antecedents, even when the antecedent is adjacent to the antecedent.

(6.7) *Reflexives*

(a) The famous man himself attended the ceremony.

(b) *The famous man itself attended the ceremony.

(c) The girl fixed herself a sandwich.

(d) *The girl fixed himself a sandwich.

Subjects also show difficulty in enforcing the agreement of auxiliaries in VP-Ellipsis constructions, as in (8).

(6.8) *VP Ellipsis*

(a) George was angry and so was Tom.

(b) *George was angry and so did Tom.

Finally, subjects show another kind of agreement difficulty, in this case much closer to home, considering the constructions which are difficult for them to comprehend.

(6.9) *Wh-Head Animacy Agreement*

(a) The pencil which he brought was nice.

(b) *The pencil who he brought was nice.

6.2 A Chain-Agreement Account of Asyntactic Comprehension

As I have implied, all of these constructions seem to involve difficulty with agreement. In particular, agreement between two elements of equivalent bar-level is implicated. That is, we have problems with phrase-level entities, in Wh-Agreement, Reflexive Agreement and Tag Subject Agreement, and between head-level entities, in Tag Auxiliary Agreement and VP Ellipsis Auxiliary Agreement. Interestingly, Subject-Verb Agreement does not seem to be a problem for these subjects. Note that Case-assignment does not seem to be a problem either. I have already noted subjects' abilities with passive participles. There are numerous other examples in (4). Given the role of Spec-Head Agreement in Case-assignment, it is interesting that both are spared. We can conclude that not all agreement is alike in the eyes of the grammar. All the problematic cases involve something we might loosely consider instances of anaphora, where agreement is between elements of like category as well as bar-level, neither of which is the case in Spec-Head Agreement.

Let us take, then, as a working hypothesis, the position that some sort of agreement process, which I will term 'anaphoric', is disrupted in agrammatism.[11] If we assume that, at least where movement chains are concerned, the relevant agreement is in the *core* feature, which includes as functional dependents Case, θ-role, and category features, then we can begin to see how a chain could be formed without the necessary result that the lexical, referential element of that chain receives a θ-role. That is, suppose we separate in principle, say into different modules, the licensing conditions which result in chain-formation from those which enforce agreement of chain-elements. That is, we wind up with a pair of modules, which I will call here Chain Theory and Trace Theory, such that the central principle of Chain Theory is the Chain Agreement Principle (37), and the central principle of Trace Theory is the Antecedent Government Condition (38). Our hypothesis is thus essentially that asyntactic comprehension reflects a disruption in syntactic processing which prevents the parser from bringing Chain Theory to bear on the construction of syntactic structures.

6.2.1 Anaphoric Agreement Theory

It seems clear, then, that the same deficit underlies both grammaticality judgment and comprehension problems, and the deficit has something to do with agreement. However, note that the parallel, though intuitively obvious, is hard to draw formally. It is a crucial feature of what I have termed anaphoric agreement that a clear distinction is drawn between agreement among overt, lexical elements, and non-overt elements. No two lexical elements can agree on θ-role features (or Case, but this is not so important to comprehension), whereas non-overt elements must agree on θ-role features with an overt element—that is the only way the overt antecedent can receive a thematic interpretation. In terms of the feature system sketched out here, agreement at the level of *core* features is impossible for two overt, lexical elements. Nonetheless it is agreement at this level that is crucial for our analysis of asyntactic comprehension.

First, note that to tie these agreement-based processing difficulties together at all it is highly desirable that we have Chain Theory be a special case of a more general theory of Anaphoric Agreement, which applies to chains in such a way as to tie together their *core* features, and to lexical phrases in such a way as to tie together only their *cat* features (which I take to be the argument of the *phi* attribute-function). One way of doing this would be to stipulate that agreement applies as high up in the feature hierarchy as it can. In the case of lexical elements, since a phrasal node which is [+*lexical*] must head its own chain with a unique Case and θ-role, agreement at any level higher than *cat* would be ungrammatical—each lexical element would be multiply defined for both Case and θ indices.

[11] I use this generalized sense of anaphora in much the same way it is used in, e.g., [Aoun 1985].

Another tack to take would involve splitting up the Chain Agreement Principle into several parts. The first condition would state that all anaphorically linked elements must share their *cat* features. This is a general condition that would apply beyond chain-formation to all instances of co-indexation of like elements. In addition, and in the same module, we would have a condition applying specifically to chains, which causes their elements to agree in θ-role and Case features. This might be seen as a condition guaranteeing the unique reference of a chain, perhaps, essentially an extension of the Visibility Condition sketched in the previous chapter.

It is not possible, I believe, to implement the first solution in the Licensing Grammar formalism, since it depends on the notion of agreement at 'the highest level of the feature hierarchy which would not yield an ill-formed result.' There is no way that I can see to implement such a hierarchy in a Licensing Grammar, and in particular to implement such a hierarchical search for the right level of agreement.

Suppose then that we mark anaphorically dependent elements with a new feature *index*.

$$index(X) \approx index(Y)$$

This usage is meant to reflect the traditional formal marking of anaphoric dependencies with co-indexation. Note that here no explicit value is assumed for the function—we need only mark terms as being index-equivalent. This is meant to be a generalization of chain-formation, so that where the grammar of Chapter 5 had that

$$chain(X) \approx chain(Y),$$

we would want to replace this with the equation in (10).

(6.10) $\quad index(X) \approx index(Y)$

Furthermore, we would want to be able to conclude, from (10), that (11) held.

(6.11) $\quad cat(core(X)) \approx cat(core(Y))$

Then, from (10) together with the additional fact that Y is the trace of X (or vice-versa), we would want to be able to conclude in addition that (12) held.

(6.12) $\quad \{ \; role(core(X)) \approx role(core(Y)), \; case(core(X)) \approx case(core(Y)) \; \}$

All of these conditions are presumed to be part of a single module, which we may as well term Anaphoric Agreement Theory (to keep it distinct from the theory of anaphoric locality, and the theory of spec-head agreement), at least for present purposes. As long as this is so, then we might predict that a focal brain injury could affect this module severely, leaving its neighbors more or less intact. What this means for processing is that, given a licensing parser, certain assertions will no longer be added to the parser's working description. In more general terms, the outcome of syntactic processing will be underdetermined in just those aspects which ought to be constrained by the Anaphoric Agreement module.

6.2.2 Passive Constructions and Chain Formation

Note that the scope of this module must include passive constructions. The key to the passive construction, according to standard GB accounts,[12] is the passive morpheme, in particular its ability to absorb the verb's Case-assigning properties, and its external θ-role. The desired effect is that the passive participle cannot assign Case to its complement, and assigns its external θ-role, optionally, to a *by*-phrase, or rather, the NP object of *by*. Accordingly the passive morpheme is assumed to be coindexed with the *by*-NP. To fit the generalizations we have discussed above, the resulting syntactic object must be more like a chain than like a lexical anaphoric dependency. In fact, the relation between the passive morpheme and the *by*-NP seems to be intermediate between these two possibilities. Clearly, on this theory, -*en* and the *by*-NP must share θ-roles, but, inasmuch as the *by*-NP gets Case from the preposition *by*, and -*en* absorbs, or is assigned, accusative Case by the verb, they cannot share Case.

Another possibility, not advanced in the literature, as far as I know, is that -*en* does not in fact absorb the external θ-role, but that this role is assigned as usual. The element that receives it would then be a null, Case-free but θ-marked element, licensed by -*en*, perhaps, and bound to the *by*-NP. Such an element could only be PRO or NP-trace (though not a trace of movement, since there is no question of movement to the *by*-phrase, only a question of chain-formation). As long as this null element is licensed by -*en*, it is hard to see that this theory would be distinct from more standard accounts. However, it is much more in line with the theory of chains and agreement as it emerges here, in that the null element so postulated would be phrasal level (-*en* is presumably a head), and not independently assigned Case.

The crucial observation, in any case, is that the *by*-NP is interpreted as being the external argument of the predicate, when it appears, but that its presence is optional, while the external θ-role is obligatory. Therefore the external role must be assigned elsewhere than to the *by*-phrase, which must inherit it through coindexing, and furthermore, as we have observed, this coindexing must be of the correct sort to transfer a θ-role. Therefore it must be more like chain-formation than like binding. We conclude that passive constructions will therefore be disrupted in agrammatism in the same way that movement constructions are disrupted.

[12] Cf., in particular, [Jaeggli 1986, Baker 1989].

6.2.3 Deriving the Data

Given these assumptions we can now turn to agrammatics' grammaticality judgment and comprehension performance and see if we can derive their errors. First, let us consider one of the ungrammatical stimuli in the difficult judgment tasks.

(6.13) $\quad$ **The girl fixed himself a sandwich.*

Given two nodes, which I will label n_g, for the node dominating *the girl*, and n_s, for the node dominating the reflexive, both the normal and the agrammatic are able, *ex hypothese*, to assert the following. (For simplicity of exposition, I will write $gndr(phi(cat(core(\alpha))))$ as simply $gndr(\alpha)$.)

$$gndr(n_g) \approx gndr(n_s).$$

Furthermore, n_g will inherit from its head that

$$gndr(n_g) \approx fem.$$

Finally, we assume that both subjects are able to access the lexical entry for *himself*, so that they both have that

$$gndr(n_s) \approx masc.$$

In addition to the above, the normal's description has, by virtue of our putative Anaphoric Agreement Principle, that

$$gndr(n_g) \approx gndr(n_s).$$

From such a description the normal can conclude, by substitution of equals for equals, that

$$gndr(n_s) \approx fem.$$

Therefore, once again by substitution,

$$masc \approx fem,$$

which violates the uniqueness of feature values ($\mathcal{T}$191). There is no better-formed description for the sentence, so the normal should be able to conclude that (13) is ungrammatical.

The agrammatic, on the other hand, lacks the ability to impose agreement on n_g and n_s, even though s/he should be able to coindex them.[13] Therefore no contradiction arises in the agrammatic's description, and the sentence should be judged grammatical.

If the stimulus had been grammatical, the agrammatic should judge it correctly, we predict. If no contradiction arises from the ostensibly ill-formed description of an ungrammatical description, none will arise from a grammatical one. Keep in mind that we have not posited a deficit in the deductive apparatus underlying parsing—the mill works, it merely lacks grist. Thus we predict performance around chance, with the overwhelming preponderance of errors coming on ungrammatical stimuli, and this is in fact what we see. Linebarger [Linebarger 1990] reports performance at about 64% correct on reflexives, and of the subjects reported on in the Linebarger, Schwartz and Saffran study [Linebarger 1983a], two showed 95% and 100% correct on grammatical stimuli, each showing 30% correct rejections of ungrammatical stimuli. Another subject in that study showed 100% correct on grammatical stimuli, and 0% correct on ungrammatical stimuli.

Let us turn now to comprehension performance. Take, for example, a passive sentence like (14).

(6.14) *The boy was chased by the girl.*

Let us refer to the node dominating *the boy* as n_b, and its trace in object position as n_t. Once again, both have that

$$index(n_b) \approx index(n_t).$$

[13] Evidence for the preservation of binding comes from work by Grodzinsky, Wexler, Chien & Marakovitz [Grodzinsky 1989]. They report near perfect performance on reflexives, which is the crucial condition for our purposes here. The task tested only the locality conditions on reflexives, and so should be taken as evidence that enough information is present in the description to subject the reflexive and its antecedent to Binding Condition A. Therefore they must at least be coindexed.

In addition, both have that

$$n_t \approx trace(n_b),$$

and that

$$role(n_t) \approx theme.$$

By virtue of the Anaphoric Agreement module, in particular its conditions on coindexed elements which are traces, the normal's parser can assert that

$$role(n_b) \approx theme.$$

The agrammatic's parser, which cannot make use of the Anaphoric Agreement module, cannot draw this conclusion. Therefore, in the agrammatic's parser's output description, the features *role* will not have a fixed value for n_b. Note that this is so even though the parser can conclude that

$$foot(chain(n_b)) \approx arg1(chase).$$

It is safe for us to assume that the structure of the chain is all there. However, if no accessible condition requires that elements of the chain share any features, then they will not share features.

The question remains as to how such partial structures will be interpreted semantically. The task to which agrammatics are subjected is usually a picture pointing task. The subject is presented with an array of pictures, and a sentence, with the object that s/he is to point to the picture which corresponds to the sentence. In particular, researchers are invariably interested in comprehension of thematic roles. Accordingly, we can think of the task as based on an elementary problem, which is to extract the thematic structure of a sentence, and the thematic structure of a picture, and compare them.

Ex hypothese, the syntax is of little or no help in this task, in constructions which involve movement. In fact, however, this is not quite true. We have assumed that the syntax fails to assign thematic roles along chains. Note, however, that if an argument remains *in situ* then we will have the following:

$$head(chain(\alpha)) \approx foot(chain(\alpha)). \quad (6.15)$$

From this it is possible to conclude that

$$role(head(chain(\alpha))) \approx role(foot(chain(\alpha)))$$

simply by applying substitution of equals for equals. That is, no special grammatical knowledge is needed to make the crucial identification here. Therefore we are justified in assuming that agrammatics correctly assign thematic roles to unmoved elements. That this is born out is indicated by a study by Caplan and Futter [Caplan 1983] (see also [Mauner 1989], for detailed discussion of these findings). In that study, objects were *in situ* in 82 stimuli, across 7 different constructions. The subject of that study assigned the thematic role to the object correctly in these constructions in all but 1 instance.

Now, consider the problem faced in interpreting the structure which subjects would compute, given one of the difficult stimuli (either a passive, or an object relative, as below).

(6.16) *The boy was kissed by the girl.*

(6.17) *The boy that the girl kissed ran away.*

If only the object has moved, then the subject should be assigned the correct role under the identity in (6.15). In that case, the semantic interpreter should be presented with a partly saturated predicate, with only its internal argument missing, and a floating NP, without a θ-role. There is only one solution to this problem: forced to guess, the semantic interpreter should treat the moved element as bearing the internal θ-role. This, of course, leads to the correct interpretation of the sentence, contrary to empirical observations of asyntactic comprehension.

We would get the right result if neither θ-role were assigned in the syntax. In such a case, the semantic interpreter, with no evidence for either θ-role assignment, would have to guess, and we would see the chance performance that we do in fact see in these tasks. On our assumptions such a result would only arise if, in constructions like (16) and (17), both the subject and object had moved. Just such a proposal exists in GB. Thus, according to numerous researchers,[14] the external argument is assigned within VP, and the subject, even of active declaratives, raises to Spec of IP, presumably to get Case. If we adopt the VP Internal Subject Hypothesis, then, we get just the right result. The verb assigns both its θ-roles to traces, and neither lexical NP emerges into the semantics with a thematic role. The best that subjects can do is to guess, as we observe them to do.

In the easy conditions, we have the expected correct assignment of the internal θ-role to the object NP, under the identity in (6.15). The subject does not get a θ-role, since it forms a (non-trivial) chain with its trace in the θ position. However, this is fairly harmless—in the semantics there is only one role left unfilled, and only one floating referent to fill it. Therefore we predict correct performance, though not as a result of correct, normal-like processing.

[14] [Koopman 1988, Speas 1985, Kuroda 1986, Kitagawa 1986].

Thus we can see that a careful, rigorous treatment of computation, and its representational basis is highly desirable in any account of truly tricky data. Otherwise it is too easy to give up a strong hypothesis! Linebarger, for example, takes agrammatics' success at judging passives as strong evidence against any kind of syntactic deficit underlying asyntactic comprehension. This, in the very same paper which she concludes by suggesting an agreement-based generalization very much like the one which underlies this chapter's investigations! Once we provide ourselves with a little logical apparatus, it becomes clear that any apparent paradox in these data just reflects the different computational and representational demands made by the comprehension and grammaticality judgment tasks.

Chapter 7

A Computational Approach to Processing Errors

The anaphoric-agreement model of asyntactic comprehension, advanced in the last chapter, predicts a certain distribution of errors according to their type. That is, this model of the deficit predicts errors on ungrammatical stimuli, but not on grammatical ones. In this chapter I will demonstrate that this is a characteristic of a much larger class of deficits, and that this class arises quite naturally in a description theoretic parser. To demonstrate this I will take here some preliminary steps towards a computational theory of language deficits,[1] a theory which attempts to come to grips with errors in computation by examining models of computation, the standards of correctness applied to such models, and ways in which computational models can deviate from these standards of correctness.

The computational approach to human language processing involves the specification of *abstract machines* which respond in some rigidly specified way upon presentation of some input. Though their metaphorical relation to actual machines is quite strong, they are, of course, mathematical objects, in particular objects of mathematical logic. Their usefulness in the study of how the mind and brain compute lies in the fact that they are mathematical, and hence transparent to investigation and possessing properties susceptible to proof in ways that the mind and brain are not.

In accord with this approach, I will proceed by examining the properties

[1] "Deficit" is too narrow a term, being most naturally understood as referring to the effects of brain damage on processing. The approach I have in mind should be relevant to any situation in which a processor is driven towards results which do not seem correct. Thus, as I will demonstrate below, this approach applies to studies of overburdened processors in normal adults, and, perhaps, processing in children with incompletely learned grammars and incompletely developed computational resources.

of some very general classes of processors, from a computational perspective. As long as our specifications of such classes are precise, we will be able to make consequences follow from them, despite the fact that we are not discussing objects with anything like the detail and complexity of, say, a working computer program let alone a human brain. In particular, I will be able to suggest reasons to doubt that the apparent paradox of the previous section is in fact paradoxical, even from the strong assumption that grammaticality judgment and parsing are tightly related processes.

7.1 Parsing, Computation and Search

There are two crucial features which define a computational problem. Every problem has associated with it a structured set (a "space") of representations which are relevant to it. For every instance of the problem (for example, for every input for which we are to compute an output) there is an associated set of solutions (maybe empty) which is a subset of the representation space. In the case of syntactic parsing the representation space might consist of phrase markers like the labeled, cross-ordered tree structures of §§2.2 & 2.3, and a solution to a particular instance of, say, the parsing problem would be a phrase marker appropriate to a given input sentence.

A *computation* of a solution involves a search of the representation space to find an element of the solution set (or perhaps the whole set). Every search procedure defines a *search space*, a sub-region of the representation space which the procedure actually examines. The procedure is given certain conditions under which it will halt. It will halt either when it has found a representation which it thinks is a solution, or when it has exhausted the search space without finding a solution. The points at which the machine will halt for an instance of a problem define the machine's solution set for that problem. The *problem's* solution set is fixed in advance by an ideal omniscient entity—these are the solutions presumed to be correct. The *machine's* solution set is determined by the procedure which drives it. Obviously, the two can differ, and the machine's response to a problem can be incorrect.

First, the solution set for any instance of the problem may lie within the machine's solution set. If it does, we say that the search procedure is *complete*. Such a procedure should be able to find a solution if there is one, but it may also halt on representations which are solutions according to its own criteria, but which are not actually solutions to the problem. A complete procedure has a solution set which is larger, or at least as large, as the problem's.

Another parameter of a search procedure's behavior is how good it is at recognizing a real solution when it sees one. A *sound* search procedure will only settle on a representation which is a solution to the problem. Nothing guarantees that it will find a solution, however. We only know of a sound procedure that, when it halts, if it tells us it has found a solution, then it has. We cannot conclude, if it halts without finding a solution, that there is no solution. It has a solution set which may be *smaller* than the problem's.

A complete but unsound search procedure will thus tend to overaccept ill-formed stimuli. In terms of the Signal Detection paradigm of psychology, it will tend to respond with False Alarms. A sound but incomplete search strategy will tend to reject well-formed stimuli; in signal detection terms it will tend to miss. A search procedure which is both unsound and incomplete will tend to respond with a combination of these errors; its solution set is disjoint from the correct set, and may overlap that set to varying degrees. Obviously, our ideal search procedure is one which is both sound and complete, that is, whose solution set exactly coincides with the problem's.

This gives us a very general way to characterize deficits in a strictly *computational* manner, according to how they affect the computations of a damaged machine. Along the lines of the preceding discussion, we might characterize an abstract deficit according to its tendency to increase or decrease the machine's solution set with respect to that of the problem.[2] We should expect that there will be numerous sub-types of deficit which each lead to one or the other effect, and we might further expect that combinations of these two effects might be possible, in which case the processor's solution set will be strictly incomparable to the problem's: larger than the problem's along some dimensions, and smaller along others. However, in this primarily exploratory research, I will ignore this more complex problem and focus on examples of the distinct effects.

Pursuing the computational approach, we model the deficit as a (minimal) distinction between two abstract machines, doing so in such a way that the abnormal device shows in its behavior something approaching the phenomena we see in clinical and experimental data. This is not quite the same thing as devising a model of the normally functioning processor and then 'breaking' it—in the first place, we want to assure ourselves that in both cases (the normal and the abnormal) we have a functioning device, a device which will produce some sort of output, no matter how degraded by

[2] I assume here that no deficit fundamentally alters the representation space of the problem. This assumption seems tenable as long as we allow that fragments of representations themselves belong to the representation space. In the case of syntactic processing, it makes a difference whether we take as the representation space the set of tree structures or the more general set of LCO structures as defined in Ch. 2, which contains the trees and also all sorts of fragments of trees.

comparison to the normal device. Of course, the demands of explanation require that the abnormal machine be recognizable as a damaged version of the normal machine. Adopting the parsing-as-search assumption, I have characterized the difference between the normal and abnormal machines as a relation between their solution sets. This allows us to use set theoretic techniques to study the difference in their behaviors. Referring to the normal parser's solution set as S_n, and the abnormal parser's space as S_a, we can restate this as follows. Either $S_n \subseteq S_a$ or $S_a \subseteq S_n$.

7.1.1 Picture Pointing and Grammaticality Judgment

To test a subject's comprehension ability, researchers typically employ a picture pointing task. The subject is presented with a sentence and a set of pictures and (in the better studies) with an additional "none of the above" option. The task is to match the sentence to the picture. In fact we can be more specific. In studies of asyntactic comprehension it is most often the thematic structure of the sentence which is being probed, that is, the subject's appreciation of who did what to whom. For purposes of exposition, let me define the *thematic structure* of a sentence, or of a picture, as a representation matching actors to their roles in a scene. Thus given a sentence like *the boy was hit by the girl* the thematic structure would be a representation in which the boy is the *undergoer* or *theme* of the action and the girl is the *agent*. Keep in mind that I am not arguing for a separate mental representational level of thematic structure—I introduce it here purely as a tool which makes it easier to analyze the logic of the picture pointing task. Presumably what I refer to as "thematic structure" could be seen as abstracting some of the aspects of a Jackendoffian Conceptual Structure [Jackendoff 1983]. Pursuing this device, we could equally well define the thematic structure of a picture in which a boy is hitting a girl to be a representation in which the boy is associated with the agent role, and the girl with the theme role.

We can now define the *picture pointing problem* in the terms laid out in the previous section. The representation space for the problem will be a set of pairs of sentences and pictures. We define a solution to the problem to be a pair in which the thematic structure of the sentence is identical to the thematic structure of the picture.

The core of the picture pointing task is thus a judgment problem, the problem of determining whether a given sentence-picture pair is or is not in the solution set. The picture pointing task as it is normally administered just iterates this core task over a set of pictures. Abstractly, we can say a subject performs sentence-picture matching judgments until s/he finds a solution, and then s/he points to that picture.

Let us ignore for the moment the more complex picture-selection task, and focus on the picture-judgment sub-task. So far subjects' ability to extract the thematic structure of pictures has not been at issue. I will assume that they are essentially correct in that task, and differ from normals only in the thematic structure extracted from the sentence. Failure on such a task is represented as either selecting a picture with the wrong thematic structure or failing to accept a picture with the correct structure. The first error is unsound, the second incomplete. Note that, if this is a correct account of the structure of the picture-judgment task, then the picture-pointing task, in a certain version, can tend to lose information about the nature of a subject's deficit. If the task is administered with a collection of pictures, and without a "none of the above" option, then both underlying errors will lead to the same performance: the selection of an incorrect picture. If however the researcher administers either a pure picture-judgment task or the more demanding picture-pointing task, with a "none of the above" option, then the two types of deficit are dissociable.

The structure of the grammaticality judgment problem is similar. Here we might think of the representation space as a set of pairs of sentences and syntactic structures. The solution set contains just those pairs for which the structure is a well-formed analysis of the sentence.[3] Errors once more come in the usual two varieties. An unsound search procedure will tend to identify as solutions pairs where the structure is not (to a normal) a well-formed syntactic analysis of the sentence, and an incomplete procedure will tend to fail in recognizing certain pairs where the structure is well-formed.

7.1.2 Information Processing

Another fairly standard assumption is that language processing involves converting information at one level of representation (e.g. the phonological) to information at another level (e.g. the syntactic), which in its own turn informs a higher level of processing (e.g. the semantic). This is a simple assumption, almost painfully so, but we will see that if we are truly committed to a computational approach to cognition, then even such a seemingly innocent assumption provides usable consequences which influence the ways in which we interpret behavioral phenomena as data. The assumption that parsing is information processing is, of course, only significant to us if we can

[3] In fact, to account for relative judgments of sentences both of which are ill-formed, we might want to say that the representation space is actually a set of triples, of sentences, structures and relative well-formedness values. Then the solution set is the set of triples whose values fall above some threshold. In the event that one is only asked to decide between fully well-formed and not fully well-formed these two versions converge.

make generalizations about deficits to information processing devices. As it happens we can once again divide up the universe of deficits, this time as they affect the flow of information through a processor. A deficit might lead to a loss of information, if we think of the deficit as increasing the noise in the system, and it might lead to the introduction of spurious information, if we model the deficit as a "short circuit", leading to "informational hallucination". As with the first example, we can combine these abstract deficits to derive a more complex picture in which both effects appear, but once again I will ignore these complexities in this dissertation. In fact I will restrict my attention even further, and consider only the effects of a loss of information. This possibility is more constrained, since all missing information "looks the same", i.e. like nothing at all, whereas at this level of generality there are few principled limits on the kinds of information which might be spuriously introduced.

Hypothesis 1 *A modeling hypothesis: Agrammatics' deficits (whatever they are) cause a loss of information, subverting an otherwise intact syntactic processor.*

In the next section we will see how these two assumptions, of search and of information processing, interact, as we examine the consequences of the information-loss model of Hypothesis 1 in what follows.

7.2 Judging and Comprehending Partial Descriptions

Making the consequences of Hypothesis 1 precise requires that we be precise about information and information processing. In particular we must be able to establish an ordering relation among representations so that we can say "representation R_1 is *more informative* than representation R_2," or that R_2 is the result of removing information from R_1.

I have assumed throughout this dissertation that the parser is constructing a logical description, as it was defined in Chapter 2: a set of declarative statements concerning what types of constituents are present in the syntactic analysis, their hierarchical dominance relations and linear precedence relations. Recall that we can order such representations according to whether one is an instance of another. In effect we can take a structure Γ which contains all the information contained in another description Δ, and maybe more, and order these two descriptions with respect to each other, saying "Γ is at least as informative as Δ," just in case Γ is an instance of Δ. In what follows I will write this relation as in example (1a).

(7.1) $\Delta \sqsubseteq \Gamma$: "description Δ is less or equally informative/restrictive than description Γ."

Information is, of course, always information *about* something. In terms of classical information theory, information is used to distinguish one message from another—it is information *about* a message. Here it is information about syntactic structures. We can thus look at the representation which the parser is building in either of two ways: either as the information structure, that is, the description itself, or as the set of syntactic structures which that information is about. Thus the fact that our first-order deductive system of Chapter 2 comes in handy here. We assume that the parser is constructing a logical description, but we can discuss its activity in terms of sets of trees where this gives us a better intuitive grasp of the problem. I introduce the following bit of notation to allow us to refer to the set of maximal extensions of a description conveniently.

(7.2) $[\![\Delta]\!]$: "the set of structures described by a (logical) description Δ." Equivalently, the set of maximal extensions of Δ.

Under ideal conditions, by the end of the parse, the description/information structure should describe/be about a single syntactic structure, namely that one which the grammar, the abstract relation between sentences and structures, pairs with the input. That is, the parse proceeds from a description with a very large set of extensions by gradually narrowing this set until, ideally, it becomes a singleton. This narrowing effect of the parse is given in formulaic form in example (3), and graphically in figure 7.1. (In what follows I will adhere to the following graphical conventions. Descriptions will be drawn with ellipses, and the sets of trees which a description describes will be drawn with boxes.)

(7.3) *Narrowing Information Structures*

$$\Delta \sqsubseteq \Gamma \rightarrow [\![\Gamma]\!] \subseteq [\![\Delta]\!]$$

("If Γ is at least as informative as Δ then Γ describes no more things than Δ." Alternatively, "Γ rules out more things than Δ does.")

It is important to keep in mind, in thinking about comprehension and grammaticality judgment, that the linguist's grammar, both syntax and semantics, is defined in terms of syntactic structures, not descriptions. The grammar is meant to represent reality for an ideal speaker/hearer, and the objects of its knowledge are similarly idealized. Here, however, we are concerned with processing which depends upon knowledge of these ideals, but

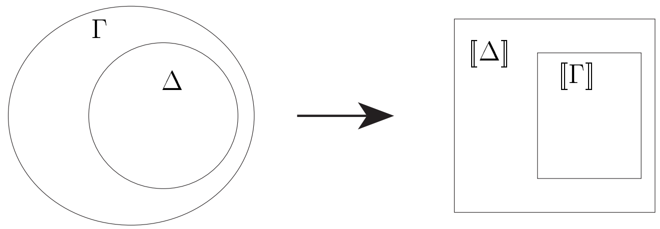

Figure 7.1: Narrowing Information Structures

which is at the same time vulnerable (in our case terribly so) to all kinds of un-ideal accident. It is the goal of linguistic aphasiology to state the effects of such accidents *in terms of* the ideal state of affairs represented in the grammar.

Consider the problem of modeling comprehension, in particular the problem we would face under Hypothesis 1 if we lacked a means of assigning a linguistic interpretation to partial descriptions. As it relates to the grammar, comprehension involves assigning a syntactic structure to a sentence and then interpreting that structure semantically. However, the description actually arrived at, following our Hypothesis 1, is not sufficiently informative to pick out just the syntactic structure which the grammar under ideal conditions would pair with the sentence. Yet what we know about semantic interpretation is stated in terms of syntactic structures, not in terms of partial descriptions. We are faced with the problem of relating a partial description to the idealized model of semantic interpretation of (ideal) syntactic structures.

The problem we face in modeling abnormal grammaticality judgment is similar. Grammaticality is normally taken to be a property of syntactic structures, namely the degree to which they satisfy the constraints which make up the linguist's grammar. If we wish to bring to bear the concepts of formal linguistics we must find some way of relating this idealized model of grammaticality to the modeling of grammaticality judgment behavior, in particular on the part of a subject *ex hypothese* unable to arrive at such a fully specified structure.

Given our modeling hypothesis, the simple formula of example (3), together with an appropriate definition of what partial descriptions mean (as in §2.4), gives us a handle on this problem. In place of Γ and Δ of example (3) we substitute the description arrived at by the normal (call it Δ_n) and the description arrived at by the aphasic subject (call it Δ_a), giving the formula in example (4), illustrated graphically in figure 7.2.

(7.4) *The Effect of Hypothesis 1*

$$\Delta_a \sqsubseteq \Delta_n \rightarrow [\![\Delta_n]\!] \subseteq [\![\Delta_a]\!]$$

> (If the normal's description (Δ_n) is more informative than the agrammatic's (Δ_a), then the normal's description describes fewer things than the agrammatic's.)

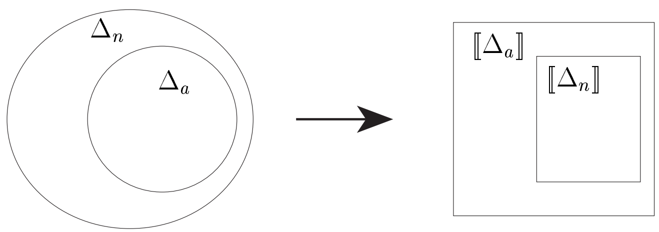

Figure 7.2: The Effect of Information Loss

Both the grammaticality judgment and comprehension tasks (in particular the picture-pointing task, but others as well) require the selection of a single interpretation, a single maximal extension, from the set $[\![\Delta_a]\!]$. Based on example (4) and Fig. 7.2, we can see that the agrammatic may select an interpretation which a normal would not select, though s/he may also select one which the normal would select. Since we are talking about specifically *syntactic* information, then the agrammatic may assign a structure to a stimulus sentence which a normal would not assign. Where comprehension depends on syntactic structure, then, we predict that the agrammatic would be liable to select the wrong semantic interpretation just where they select the wrong syntactic interpretation. We predict further that, as long as the deficit leads only to a loss of syntactic information, subjects' comprehension performance will not be otherwise impaired.

These are not in themselves particularly startling predictions. Any account which posits a damaged syntax will derive the result that comprehension will be impaired where it depends on syntactic structure. (Though it is

not at all trivial to say in such a case how comprehension can proceed at all!) The problem arises in the treatment of grammaticality judgment— somehow we must impair syntactically-determined semantic interpretation but leave the assessment of syntactic well-formedness intact.

Grammaticality judgment is in fact a scalar judgment, but with a definite endpoint, namely *fully grammatical*. A model of grammaticality judgment which is true to the linguist's conception of grammar and grammaticality is thus not strictly speaking set theoretic, but involves some rather more complicated notions. However, in neurolinguistic practice grammaticality judgments are usually presented as binary judgments, of either "fully grammatical" or "less than fully grammatical." Therefore we can continue to treat these problems set theoretically, which has the advantage of being intuitive and easy to visualize. Accordingly, we treat the syntactic structures which are fully well-formed as a set determined by a particular grammar. Call the grammar G, and the set of fully well-formed structures $\mathbf{L}(G)$.[4] In the our framework of §7.1.1, the set $\mathbf{L}$(G) is the solution set to the grammaticality judgment problem. We can then characterize (binary) grammaticality judgment as a process involving the construction of a syntactic representation of the stimulus, and the determination of whether or not the resulting (stimulus, representation) pair is a member of the set $\mathbf{L}(G)$. Note that this is just a characterization of the *problem*, not the specification of an algorithm for *solving* the problem. It should be quite clear that determining membership in $\mathbf{L}(G)$ does not necessarily involve leafing through a list of its members until one either finds the structure or come to the end of the list. By the same token, the application of parsing algorithms such as that sketched in Chapter 3, which construct descriptions directly from information in the string, does not in any way alter the characterization of the problem as a search problem.

The grammar G which generates $\mathbf{L}(G)$ typically involves constraints on the form of trees. In order to assess the consequences of our modeling hypothesis, we need a characterization of grammaticality judgment which applies more generally to partial descriptions of trees. As long as we have a way of interpreting partial descriptions as sets of trees, e.g. via the maximal extensions of Chapter 2, this characterization is straightforward, and yields as a special case the classic characterization, where it is individual trees that are

[4] It is usually the case that a *language* is defined as a set of strings *generated* from a grammar, and this set of strings/sentences is written $L(G)$. I am using the boldface **L** here because we are speaking of a set of *structures* rather than of *sentences*. The relation between the two is fairly clear, since we could construct the set $L(G)$ from the terminal, "leaf nodes" of the structures in $\mathbf{L}(G)$. Note, however, that the two sets are not isomorphic, since several different structures can yield the same string, whenever the language allows structural ambiguities.

judged.

Def'n 4 *A logical description Δ is grammatical just in case*

$$(\llbracket \Delta \rrbracket \cap \mathbf{L}(G)) \neq \emptyset.$$

That is, Δ is grammatical just in case one of the trees which it describes is fully well-formed.

Definition 4 is not entirely trivial. Note that we could just as well have insisted that *all* the trees described by Δ be fully well-formed. It is not completely clear how to decide this issue. However, it seems relevant that in judging structurally ambiguous sentences, such as *I was watching a man with a telescope*, we appear not to make ourselves consider all of its structural analyses, but are content that it is well-formed on whichever reading we happen on first.

Figure 7.3 is a graphic depiction of the formula in Def'n 4. The illustration in figure 7.3 represents a situation in which one or more readings of a sentence have well-formed syntactic representations.

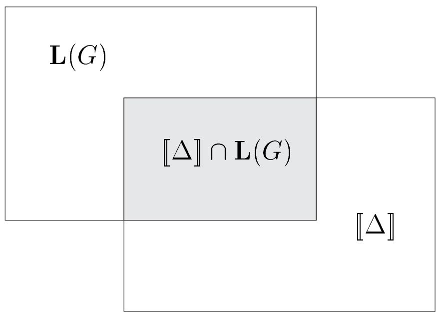

Figure 7.3: Grammaticality Judgment—Stimulus Grammatical

Now consider what happens when we combine our model of grammaticality judgment in Def'n 4 and figure 7.3 with our model of an information loss deficit in example (4) and figure 7.2. The result is depicted in figure 7.4.

The dark grey region in figure 7.4, labeled $\llbracket \Delta_n \rrbracket \cap \mathbf{L}(G)$, represents the fully grammatical readings of the sentence which fall within the normal's search space. Because the agrammatic cannot narrow down his description as far as the normal can, *ex hypothese*, there are at least as many structural

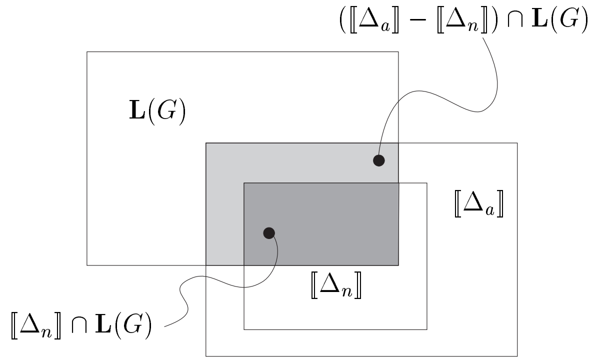

Figure 7.4: Normals vs. Agrammatics—Stimulus Grammatical

analyses accessible to him as to the normal. Thus in every case if there is a fully well-formed reading available to the normal, there is a fully well-formed reading available to the agrammatic.

Now consider the light gray region in figure 7.4. This represents the fully well-formed readings which may fit the agrammatic's looser description without fitting the normal's tighter one. We have no way of knowing *a priori* which analysis the abnormal subject will settle on. It may be one which a normal might agree to, or it may not. Whenever the abnormal subject selects an analysis which a normal would not select, we have the following situation. The abnormal still judges the sentence to be grammatical, *but for the wrong reason*. If asked what the sentence means, the abnormal subject will provide a semantic interpretation which a normal would not derive from that sentence. Thus we predict, for grammatical sentences, that an agrammatic subject will perform like a normal on a grammaticality judgment task, but sub-normally on a comprehension task.

We turn next to ungrammatical sentences, sentences for which no syntactic analysis falls within the set of fully well-formed structures. This condition is illustrated in figure 7.2.

Once again, we take this as our model of normal grammaticality judgment, given an ungrammatical stimulus, and combine it with our model of the relation between normal and abnormal parsing. The result is illustrated in figure 7.2. Here we see a potential divergence in grammaticality judgment performance between the normal and the abnormal. Because the looser description arrived at by the abnormal parser allows more potential readings than that which the normal constructs, it is possible that, while none of

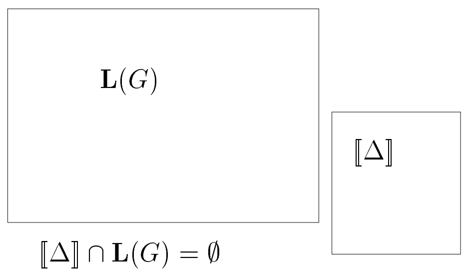

Figure 7.5: Grammaticality Judgment—Stimulus Ungrammatical

the normal's analyses are fully well-formed, some of the abnormal's analyses are. In such cases, the subject will return a judgment of "grammatical" even though the stimulus is ungrammatical.

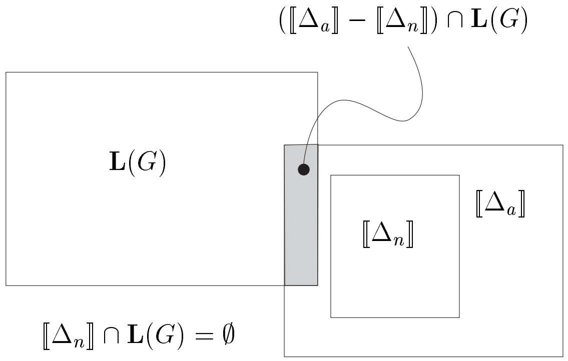

Figure 7.6: Normals vs. Agrammatics—Stimulus Ungrammatical

Thus we can see what happens when we take some very general formal characterizations of parsing and grammaticality judgment, and combine them with an equally formal (and equally general) computational characterization of a functional deficit, namely the loss of information. Such a model predicts normal-like grammaticality judgment performance but abnormal comprehension performance while stimuli are grammatical, and here it is in accord with the facts as presented earlier. However, it also predicts abnormally high acceptance of ungrammatical stimuli, which does not seem at first glance in accord with the reports which lead to the syntax-there-but-not-there paradox.

Looking more closely at some of the relevant articles, however, we find quotes like the following.

> "...in three of the four subjects the tendency is to err by accepting ill-formed sentences." [The fourth subject had an absolutely consistent pattern of judging all tag-question stimuli ungrammatical.—tlc] (from Linebarger et al. (1983a))

> "In both the analyses by subjects and by items there was a significant effect of grammaticality...It proved to be easier for the subjects to give a correct judgment when the correct inflections were presented." (from Lukatela et al. (1988))

Similarly Shankweiler et al. [Shankweiler 1989] found that for within-class substitutions of Prepositions, Particles, Determiners and Auxiliaries, subjects had mean scores of 61% correct, as opposed to 80% correct on good stimuli. Subjects remained highly sensitive to between-class substitutions, which boosted the average score, on which the authors based their discussion. We conclude that the predictions made by this information-loss model are indeed in line with the facts as they have actually been reported in the literature.

Before we go on to consider an alternative deficit model, it is worth pausing a moment to consider what a model like this cannot do. Because it is a logical model, it does not furnish numerical predictions, and in particular it cannot furnish probabilistic predictions. It is widely recognized that the effects of deficits can be quite variable even from day to day within the same subject. Therefore testable predictions have always to be of the form "subject will *tend* to perform badly on task condition X," rather than "subject will fail on task condition X." This is a subtle distinction, to be sure, but one which has tended to delay the acceptance of logical models of language breakdown by the neuropsychological community. I would argue that the objection is ill-founded. Given that we wish, pending a full understanding of the factors determining human linguistic performance, to introduce an element of non-determinism into models of language breakdown we have, no matter what the basis of our model, to introduce some sort of random number generator into our model of the deficit (cf. [Haarman, in press] for just such an example). Yet it should be clear that subjecting the predictions of an otherwise deterministic model to modulation by a random filter does not in any way alter the aspects of the model which are in fact enlightening. Surely there can be no interest in the random number generator itself, in particular as it is just a stand-in for processes which are complex to be sure but hardly random! Therefore I take the failure of this model to predict fluctuating behavior to be a less than compelling objection, though it is certainly a factor to consider in fitting its predictions to data.

Another striking fact about agrammatics' performance is that although they have significant trouble with ungrammatical stimuli, they generally perform well above chance, whereas in comprehension tasks subjects showing asyntactic comprehension are reported to perform at chance. These differences most likely reveal an effect of task complexity. It has been shown that judgment is an easier task than either the picture-pointing or act-out tasks which are usually used to measure comprehension (cf. [Crain 1985]). Such an explanation depends on a deficit whose effect is exacerbated by task complexity, and that effect should be carried over to a model of the deficit. In our terms, it must be the case that when processing is more difficult, then proportionally more information is lost. This is a more serious problem, but it cannot really be addressed without an independent model of the sources of task complexity, the reasons why a task like a multiple-choice picture pointing task makes more demands than a simpler judgment task. The development of such a model clearly falls outside the domain of linguistics, however, and I will accordingly leave it as an open question.

Another way in which a model like this can be sharpened depends on a more detailed appreciation of linguistic theory. Here I have made no attempt to offer a linguistic characterization of the deficit underlying the pattern of behavior under discussion. As a result this model cannot make predictions about which particular sentences subjects will have difficulty with. In this dissertation I am chiefly concerned with the way deficits affect how an abstract machine computes, and not with the details of any particular such machine. I am concerned with *how* the processor breaks down, rather than *where*. However, it is a fairly straightforward matter to take a particular model of the grammar, or a functional description of the processor, and propose a site at which information is lost in that model. Then, combined with the logical, computational approach presented here, such a model should make predictions about performance on particular grammatical constructions. (The series of papers [Cornell 1989, Cornell 1990, Mauner 1990] presents several such deficit-location hypotheses, coupled with earlier versions of the deficit-type analysis presented here.)

7.3 Loss of Processing Resources—An alternative hypothesis

In the previous section we considered a class of computational deficits which tends to increase the parser's solution set. Here we will sketch a class of deficits which tends to decrease its search space (and hence, potentially, its solution set).

The performance of a computational device is generally measured by computer scientists in its use of space (working memory, buffer space, etc.) and its use of time. A more efficient device will use less time and less memory in order to arrive at a solution. This gives us an extremely general handle on the notion of *processing resources*, and on the related notion of the *loss* of processing resources. In this section we will examine the effect on the performance of a computational device of the loss of space and/or the loss of time. By "loss of space" we mean a deficit which removes from use some portion of the subject's working memory, the memory used to hold intermediate results of processing and to explore alternative analyses of the input. By "loss of time" we mean something similar, a deficit which either retards the process of constructing intermediate results or speeds the process of their decay, of their removal from working memory (cf. [Haarman, in press]). One way or another, we hypothesize a loss in the ability of the processor to support and sustain its intermediate results during the processing task.

Hypothesis 2 *Another modeling hypothesis—Agrammatics' deficits (whatever they are) cause a loss of processing resources.*

The effects of limiting the parser's resources have already been investigated in computational research on the processing of garden path sentences in normals. By strictly limiting the size of a "lookahead buffer" in a deterministic parser, [Marcus 1980] showed how the parser could be prevented from arriving at an analysis of certain "garden path" sentences such as

(7.5) *The horse raced across the meadow fell.*

These sentences are generally believed by linguists to be grammatical (cf. *The tractor driven across the meadow stopped*), but very often normal subjects judge them ungrammatical. Descriptively, they are led down the "garden path" by a verb form which looks so much like a past tense form (*raced* in this example) that its correct analysis, as the passive participle of a "reduced relative clause", is ignored. It is in fact ignored so completely that in many cases only once the "trick" has been carefully explained will a subject "get it". A similar effect is found with multiply center embedded sentences like (6), and sentences with highly ambiguous lexical items like (7) and (8).

(7.6) *The man the boy the girl likes saw outside was a thief.*

(7.7) *Buffalo buffalo buffalo buffalo buffalo Buffalo buffalo.*

(7.8) *Police police police police police.*

Like garden path sentences, these examples, though apparently grammatical, and detectably so with enough help from pencil, paper, and kindly disposed people who know the correct analysis, are often judged ungrammatical.[5] Furthermore most accounts of such sentences invoke the insufficiency of the processor's parsing resources, typically some sort of working memory. (Cf. [Wu 1989], and refs. for a review of the literature on center embedding.)

Computationally, the effect of strictly limiting resources in this way is to restrict the search space accessible by the parser. Certain solutions which are particularly expensive to compute (in the garden-path case, which require a longer lookahead in order to be processed deterministically) now fall outside the shortened search space. If we assume that grammaticality judgment involves the search for a well-formed analysis of the input, and that the failure to find such an analysis leads to a judgment of ungrammaticality, then the garden path phenomenon is potentially accounted for under such an analysis. A well-formed analysis of the input exists, in these cases, and subjects can be led to it by the appropriate use of top-down constraint, which fixes the desired interpretation in advance and so suppresses the tempting garden path analysis. However, under normal circumstances, subjects' unaided parsers cannot find this analysis, and so they judge the sentence to be ungrammatical.

It should already be clear what our predictions are if we apply Hypothesis 2 to agrammatics. A graphic illustration of the model is given in Figure 7.7. In the figure, the point labeled S_1 represents the analysis of an ungrammatical sentence. The point labeled S_2 represents a fully grammatical sentence with a relatively resource-inexpensive analysis. The point labeled S_3 represents a fully grammatical sentence with a relatively resource-expensive analysis.

[5] The "buffalo" examples (one can construct a vast number of comparable examples) depend on the ambiguity of the word *buffalo* as being either the noun meaning bison, the adjective derived from the city in western New York state, or the verb meaning roughly "to confuse". Thus the sentence in (7) is several ways ambiguous, but can mean among other things something like "Bison from Buffalo, NY, whom bison baffle, (in turn) baffle (other) bison from Buffalo, NY." The "police" examples are based on relative clauses and the lexical ambiguity of *police* as either a noun or the verb referring to the activities of police. Thus the sentence in (8) can mean "Police whom (other) police keep an eye on, (in turn) keep an eye on (other) police."

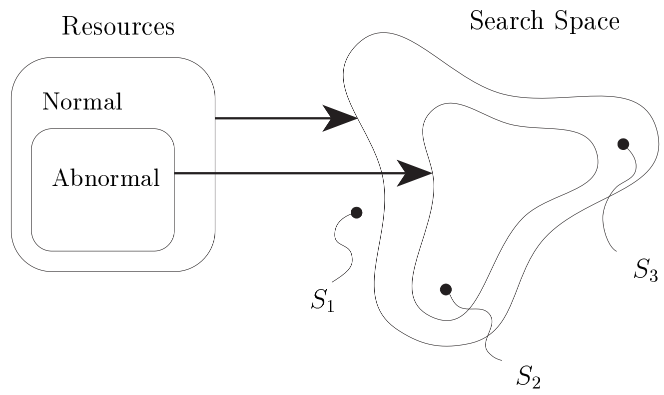

Figure 7.7: The Resource Loss Hypothesis.

As Figure 7.7 illustrates, if we present subjects suffering from a resource-loss deficit with an ungrammatical sentence, they should still judge it ungrammatical. If the sentence really is ungrammatical, then no amount of extra resources should make it grammatical, and still less a decrease in available resources. If we ask for judgments of grammatical sentences, on the other hand, then subjects' performance diverges. Normal subjects should not distinguish between sentence S_2 and sentence S_3, barring examples like garden path sentences, or multiply center-embedded constructions. Abnormal subjects, on the other hand, should be hindered by the extra complexity involved in processing sentence S_3 from discovering its correct well-formed analysis. As a result, they should judge such a sentence ungrammatical where a normal would judge it grammatical.

Thus we see the opposite pattern of performance on a grammaticality judgment task, if we adopt Hypothesis 2, from that predicted by Hypothesis 1. Subjects suffering from a pure *information loss* deficit should differ from normals in their response to *ungrammatical* stimuli, while subjects suffering from a pure *resource loss* deficit should differ from normals in their response to *grammatical* stimuli.

The effects of a resource loss deficit on comprehension are a little harder to work out, for the following reason. The grammaticality-judgment problem involves constructing a syntactic representation and evaluating the success of that construction. The picture-judgment problem involves constructing a syntactic representation, and then further extracting the thematic information from that structure, together with the lexical semantics of the predicate.[6] Then the thematic structure (however that is in fact represented) is judged against the thematic structure of the picture. Thus we have to assess the effects of the deficit on processing which takes place *downstream* from the presumed location of the deficit. This was a fairly straightforward task under Hypothesis 1, since we had a well-defined notion of what structure was actually computed under that deficit. It is not so clear what structure is assigned by a parser with shortened resources. I assume that *some* structure is assigned. We can, for example, often understand sentences that have no fully well-formed structure—it would seem that we get *something* from them.

(7.9) **Albert likes the picture that Waldo sold it to him.*

I will assume, then, that parsing is a sort of optimization problem, the problem of finding the best-formed analysis of a sentence, even if the best that can be done still falls short of full well-formedness. This is a far more sophisticated conception of a parser than I can suggest an implementation for, but the consequences of such an assumption are clear enough that we can discuss it at this level of generality. One way in which a structure can deviate is to be fragmentary, and without well-formed extensions. Thus it is often claimed as a benefit of bottom-up parsing algorithms that, even if the chunks constructed cannot be knit together to form a well-formed structure, nonetheless at least the parser can pass on the collection of chunks, leaving it up to higher-level processes to sort them out.

Such a conception fits well within the framework of this dissertation—the parser constructs a partial description (with no well-formed extensions). Then the semantic and pragmatic interpreters which take that description as their input might be expected to respond in some way which was constrained by syntactic information, where there was any, and they would fill in the gaps, in a manner analogous to generating maximal extensions at the syntactic level, where no structural information was supplied. If this were so it would give us an interesting relationship between information loss and resource loss deficits: as we go downstream from the site of a resource loss deficit, its effects look more and more like information loss. Accordingly, we might expect comprehension performance to be the same regardless of deficit type. The "dynamic" approach to deficit modeling pursued by Kolk

[6] Thus, for instance, the syntax of the following two sentences is identical, but the different predicates assure that the subject and indirect object will have their roles reversed.

1. *John bought a car from David.*

2. *David sold a car to John.*

and his colleagues (see [Kolk 1985, Haarman, in press], and cf. [Gigley 1983]) is of this sort. In particular in [Haarman, in press], the deficit is a resource deficit, affecting the amount of time which the parser has to build syntactic constructions. However, because the deficit is clearly one which is meant to affect syntactic processing, and because the data they are modeling are comprehension, rather than grammaticality judgment, it is difficult to draw comparisons between their approach and one which posits an information loss deficit. However, within the framework I have proposed here, such a model should predict a high rate of incorrect rejections of grammatical stimuli as the dominant error type on such a task.

If, on the other hand, the deficit is located closer to the locus of semantic interpretation, then we do predict different effects, at least on a picture judgment task. Linebarger et al. [Linebarger 1983a, Schwartz 1985] propose such a deficit to account for asyntactic comprehension in the face of apparently preserved grammaticality judgments. I have argued above that grammaticality judgment is in fact disrupted, which we discover only once we have reason to focus on errors on ungrammatical sentences only. However, assuming for the moment that there are subjects with uncontroversially preserved judgment abilities, we might want to adopt the Mapping Hypothesis which these researchers propose.

According to such a view, the deficit would be to the procedure which identifies the correct thematic structure corresponding to a particular syntactic structure. If this were a resource loss deficit, then we would expect, on a simple picture judgment task, the same sort of pattern we predict from a syntactic resource loss deficit and a grammaticality judgment task, namely the subject will have problems when the sentence does in fact match the picture, and not when it does not.

This very general approach to processing difficulties is too general to lend much positive support to a theory as specific as the anaphoric agreement model of the previous chapter. However, by the same token, it rules out the very large class of syntactic resource loss deficits which could be advanced to account for asyntactic comprehension with (mostly) preserved grammaticality judgments. Thus it would be wrong to dismiss such a coarse-grained analysis as this simply because it was coarse-grained. Given the huge search-space of possible deficit theories, these coarse-grained theories give us the leverage we need to remove large sub-spaces in one fell swoop, allowing us to focus on an evidently more profitable, and smaller, remaining space of possible theories.

Bibliography

[Aoun 1979] Joseph Aoun (1979) On government, Case-marking, and clitic-placement. Unpublished ms., MIT.

[Aoun 1985] Joseph Aoun (1985) *A Grammar of Anaphora*. Cambridge, MA: MIT Press.

[Abney 1986] Steven P. Abney & Jennifer Cole (1986) A Government-Binding Parser. In the proceedings of *NELS 16*, pp. 1-17.

[Abney 1987] Steven P. Abney (1987) Licensing and parsing. In the proceedings of *NELS 17*, MIT.

[Abney 1989] Steven P. Abney (1989) A Computational Model of Human Parsing. *Journal of Psycholinguistic Research 18*, pp. 129-144.

[Aho 1972] Alfred V. Aho & Jeffrey D. Ullman (1972) *The Theory of Parsing, Translation, and Compiling (Volume 1: Parsing)*. Englewood Cliffs, NJ: Prentice-Hall.

[Baker 1989] Mark Baker, Kyle Johnson & Ian Roberts (1989) Passive Arguments Raised. *Linguistic Inquiry*.

[Barton 1985] G. Edward Barton & Robert C. Berwick (1985) Parsing with assertion sets and information monotonicity. In the proceedings of *IJCAI-85*, pp. 769-771.

[Berwick 1987] Robert C. Berwick (1987) *Principle-based parsing*. AI Tech. Report #972, MIT.

[Berwick 1991] Robert C. Berwick (1991) Principle-Based Parsing. In P. Sells, S. M. Shieber & T. Wasow (Eds.) *Foundational Issues in Natural Language Processing*. Cambridge, MA: MIT Press.

[Berwick 1985] Robert C. Berwick & Amy Weinberg (1985) Deterministic Parsing: A modern view. In the proceedings of *NELS 15*, pp. 15-33.

[Bouchard 1984] Denis Bouchard (1984) *On the Content of Empty Categories*. Dordrecht: Foris.

[Bradley 1980] Dianne C. Bradley, Merrill E. Garret, Edgar B. Zurif (1980) Syntactic deficits in Brocas aphasia. In David Caplan (Ed.) *Biological Studies of Mental Processes*. Cambridge, MA: MIT Press.

[Caplan 1983] David Caplan (1983) Syntactic competence in agrammatism—a lexical hypothesis. In Michael Studdert-Kennedy (Ed.) *Psychobiology of Language*. Cambridge, MA: MIT Press.

[Caplan 1986] David Caplan & C. Futter (1986) Assignment of thematic roles by an agrammatic aphasic patient. *Brain & Lg. 27*, pp. 111-134.

[Caramazza 1976] Alphonso Caramazza & Edgar B. Zurif (1976) Dissociation of algorithmic and heuristic processes in language comprehension: evidence from aphasia. *Brain & Lg. 3*, pp. 572-82.

[Chomsky 1955/75] Noam Chomsky (1955/75) *The Logical Structure of Linguistic Theory*. Chicago: Univ. of Chicago Press.

[Chomsky 1957] Noam Chomsky (1957) *Syntactic Structures*. The Hague: Mouton.

[Chomsky 1964] Noam Chomsky (1964) *Current Issues in Linguistic Theory*. The Hague: Mouton.

[Chomsky 1965] Noam Chomsky (1965) *Aspects of the Theory of Syntax*. Cambridge, MA: MIT. Press.

[Chomsky 1981] Noam Chomsky (1981) *Lectures on Government & Binding*. Dordrecht: Foris.

[Chomsky 1986a] Noam Chomsky (1986a) *Knowledge of Language: Its Nature, origin, and Use*. New York: Praeger.

[Chomsky 1986b] Noam Chomsky (1986) *Barriers*. Cambridge, MA: MIT Press.

[Chomsky 1988] Noam Chomsky (1988) *Some Notes on Economy of Derivation and Representation*. Ms., MIT.

[Cornell 1990a] Thomas L. Cornell (1990a) A logic of descriptions: Foundations of principle-based grammars and parsers. Paper presented at the Western Conference on Linguistics, El Paso, TX. (To appear in the proceedings.)

[Cornell 1990b] Thomas L. Cornell (1990b) Logical Descriptions and Non-Construction-Specific Principles. In R. Mulder, R. Sybesma & J. Van Lit (Eds.) *Proceedings of the Leiden Conference of Junior Linguists 2*. Leiden, The Netherlands.

[Cornell 1991] Thomas L. Cornell (1991) Logical Descriptions: Deriving structure from licensing conditions. Paper presented at the 65th Annual Meeting of the Linguistic Society of America, Chicago, IL.

[Cornell 1992] Thomas L. Cornell (1992) Judging Fragmentary Representations. Paper presented at the 66th Annual Meeting of the Linguistic Society of America, Philadelphia, PA.

[Cornell 1989] Thomas L. Cornell, Victoria A. Fromkin & Gail Mauner (1989) A computational model of linguistic processing: Evidence from aphasia. Paper presented at the meeting of the Academy of Aphasia, Santa Fe, NM.

[Cornell 1990] Thomas L. Cornell, Victoria A. Fromkin & Gail Mauner (1990) A formal model of linguistic processing: Evidence from aphasia. In Aaron L. Halpern (Ed.) *The Proceedings of the Ninth West Coast Conference on Formal Linguistics*. Stanford, CA: CSLI.

[Crain 1985] Stephen Crain & C. McKee (1985) Acquisition of Structural Restrictions on Anaphora. In the proceedings of *NELS 16*, Amherst, MA.

[Enderton 1972] Herbert B. Enderton (1972) *A Mathematical Introduction to Logic*. New York: Academic Press.

[Fitting 1990] Melvin Fitting (1990) *First-Order Logic and Automated Theorem Proving*. Texts and Monographs in Computer Science. New York: Springer Verlag.

[Frank 1990] Robert Frank (1990) *Computation and Linguistic Theory: A Government Binding Theory Parser Using Tree Adjoining Grammar*. University of Pennsylvania M.A. thesis.

[Gallier 19??] Jean H. Gallier & Stan Raatz (1987) HORNLOG: A graph-based proof procedure for Horn clauses. *Journal of Logic Programming 3*, pp. 119-158.

[Gazdar 1985] Gerald Gazdar, Ewan Klein, Geoffrey Pullum & Ivan Sag (1985) *Generalised Phrase Structure Grammar*. Cambridge, MA: Harvard Univ. Press.

[Gigley 1983] H. M. Gigley (1983) HOPE-A1 and the dynamic process of language behavior. *Cognition & Brain Theory 4*, pp. 38-85.

[Goodglass 1972] Harold Goodglass & Edith Kaplan (1972) *Assessment of Aphasia and Related Disorders*. Philadelphia: Lea & Febiger.

[Grodzinsky 1984] Yosef Grodzinsky (1984) The syntactic characterization of agrammatism. *Cognition 16*, pp. 99-120.

[Grodzinsky 1986] Yosef Grodzinsky (1986) Language deficits and the theory of syntax. *Brain & Lg. 27*, pp. 135-159.

[Grodzinsky 1990] Yosef Grodzinsky (1990) *Theoretical Perspectives on Language Deficits*. Cambridge, MA: MIT Press.

[Grodzinsky 1989] Yosef Grodzinsky, Ken Wexler, Yu-Chin Chien & Susan Marakovitz (1989) The Breakdown of Binding Relations. Paper presented at The Academy of Aphasia, Santa Fe.

[Haarman, in press] Henk J. Haarman & Herman H. J. Kolk (in press) A computer model of the temporal course of agrammatic sentence understanding: The effects of variation in severity and sentence complexity. To appear in *Cognitive Science*.

[Heilman 1976] K. M. Heilman & R. J. Scholes (1976) The nature of comprehension errors in Brocas, Conduction, and Wernickes aphasias. *Cortex 12*, pp. 258-265.

[Hyams 1986] Nina Hyams (1986) *Language Acquisition and the Theory of Parameters*. Dordrecht: Reidel.

[Jackendoff 1977] Ray Jackendoff (1977) *$\bar{X}$ Syntax: A Study of Phrase Structure*. Cambridge, MA: MIT Press.

[Jackendoff 1983] Ray S. Jackendoff (1983) *Semantics and Cognition*. Cambridge, MA: MIT Press.

[Jacobson 1990] Pauline Jacobson & Mark Johnson (1990) Tree Generation with Context-Sensitive Rules Yields only Context-Free Languages. In the proceedings of *WCCFL9*, Stanford University, pp. 289-304.

[Jaeggli 1986] Osvaldo A. Jaeggli (1986) Passive. *Linguistic Inquiry 17*, pp. 587-621.

[Johnson 1988] Mark Johnson (1988) *Attribute-Value Logic and the Theory of Grammar*. CSLI Lecture Notes 16. Stanford: CSLI.

[Johnson 1989] Mark Johnson (1989) Parsing as deduction: the use of knowledge of language. *Journal of Psycholinguistic Research 18*, pp. 105-128.

[Johnson 1990] Mark Johnson (1990) Expressing disjunctive and negative feature constraints with classical first-order logic. In *28th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*.

[Johnson 1991] Mark Johnson (1991) Features and Formulae. *Computational Linguistics 17*, pp. 131-151.

[Joshi 1982] Aravind K. Joshi & Leon S. Levy Phrase structure trees bear more fruit than you would have thought. *American Journal of Computational Linguistics*, vol. 8, pp. 1–11.

[Joshi 1985] Aravind K. Joshi (1985) Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D. R. Dowty, L. Karttunen & A. M. Zwicky (Ed.) *Natural Language Parsing:*

Psychological, Computational, and Theoretical Perspectives. Cambridge: Cambridge Univ. Press.

[Kaplan 1982] Ronald Kaplan & Joan Bresnan (1982) Lexical Functional Grammar: a Formal System for Grammatical Representation. In J. Bresnan (Ed.) *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press.

[Kasper 1987] Robert T. Kasper (1987) *Feature Structures: A Logical Theory with Application to Language Analysis*. Univ. of Michigan Ph.D. thesis.

[Kasper 1986] Robert T. Kasper & William C. Rounds (1986) A Logical Semantics for Feature Structures. In the proceedings of the *24th Annual Meeting of the ACL*, pp. 257-266.

[Kim 1988] Kim Yang Soon (1988) *Licensing Principles and Phrase Structure*. University of Wisconsin, Madison, Ph.D. thesis.

[Kitagawa 1986] Yoshihisa Kitagawa (1986) *Subject in Japanese and English*. U. Mass. Ph.D. dissertation.

[Kolk 1985] Herman H. J. Kolk, Marianne J. F. van Grunsven & Antoine Keyser (1985) On parallelism between production and comprehension in agrammatism. In M.-L. Kean (Ed.) *Agrammatism*. New York: Academic Press.

[Koopman 1984] Hilda Koopman (1984) *The Syntax of Verbs: From Verb Movement Rules in the Kru Languages to Universal Grammar*. Dordrecht: Foris.

[Koopman 1988] Hilda Koopman & Dominique Sportiche (1988) *Subjects*. Unpublished UCLA Ms.

[Kuroda 1986] S-Y. Kuroda (1986) *Whether We Agree or Not*. Ms., UC San Diego.

[Lasnik 1977] Howard Lasnik & Joseph J. Kupin (1977) A Restrictive Theory of Transformational Grammar. *Theoretical Linguistics 4*, pp. 173-196.

[Lebeaux 1989] David Lebeaux (1989) *Language Acquisition and the Form of the Grammar*. Univ. of Massachusetts Ph.D. thesis.

[Linebarger 1983a] Marcia C. Linebarger, Myrna F. Schwartz & Eleanor M. Saffran (1983a) Sensitivity to grammatical structure in so-called agrammatic aphasics. *Cognition 13*, pp. 361-392.

[Linebarger 1989] Marcia C. Linebarger (1989) Neuropsychological evidence for linguistic modularity. In Greg N. Carlson and Michael K. Tanenhaus (Eds.) *Linguistic Structure in Language Processing*. Dordrecht: Kluwer.

[Linebarger 1990] Marcia C. Linebarger (1990) Neuropsychology of Sentence Parsing. In Alphonso Caramazza (Ed.) *Cognitive Neuropsychology and Neurolinguistics*. Hillsdale, NJ: Lawrence Erlbaum.

[Lloyd 1987] J.W. Lloyd *Foundations of Logic Programming*. 2nd edition, New York: Springer-Verlag.

[Lukatela 1988] K. Lukatela, Stephen Crain & Donald Shankweiler (1988) Sensitivity to inflectional morphology in agrammatism: Investigation of a highly inflected language. *Brain & Lg. 33*, pp. 1-15.

[Macías 1990] Benjamín Macías (1990) *An Incremental Parser for Government-Binding Theory*. University of Cambridge Ph.D. thesis.

[Marcus 1980] Mitchell P. Marcus (1980) *A Theory of Syntactic Recognition for Natural Language*. Cambridge, MA: MIT Press

[Marcus 1983] Mitchell P. Marcus, Donald Hindle & Margaret M. Fleck (1983) D-Theory: Talking about talking about trees. In *21st Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 129-136.

[Mauner 1989] Gail Mauner (1989) A Structural Account of Sentence Comprehension and Related Phenomena in Agrammatism. Ms. UCLA & Rochester.

[Mauner 1990] Gail Mauner, Thomas L. Cornell, Victoria A. Fromkin (1990) Explanatory models of aphasia. Paper presented at the meeting of the Academy of Aphasia, Baltimore, MD.

[May 1985] Robert May (1985) *Logical Form: Its Structure and Derivation*. Cambridge, MA: MIT Press.

[McCawley 1968] James McCawley (1968) Concerning the base component of a transformational grammar. *Foundations of Language 4*, pp. 243-269.

[Moshier 1988] Michael Andrew Moshier (1988) *Extensions to Unification Grammar for the Description of Programming Languages*. U. Mich. PhD. thesis.

[Moshier 1991] M. Andrew Moshier (1991) On Completeness Theorems for Feature Logics. To appear in the proceedings of *MOL-2*.

[Moshier 1987] Michael Andrew Moshier & William C. Rounds (1987) A Logic for Partially Specified Data Structures. In the proceedings of *ACM Symposium on the Principles of Programming Languages*, pp. 156-167.

[Partee 1990] Barbara H. Partee, Alice ter Meulen & Robert E. Wall (1990) *Mathematical Methods in Linguistics*. Dordrecht: Kluwer.

[Pereira 1980] Fernando C. N. Pereira & David H. D. Warren (1980) Definite clause grammars for language analysisa survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence 13*, pp. 231-278.

[Pereira 1987] Fernando C.N. Pereira & Stuart M. Shieber (1987) *Prolog and Natural-Language Analysis*. CSLI Lecture Notes 10. Stanford: CSLI.

[Peters 1973] P. S. Peters & R. W. Ritchie (1973) Context-Sensitive Immidiate Constituent Analysis: Context-Free Languages Revisited. *Mathematical Systems Theory 6*, pp. 324-333.

[Pollard 1987] Carl Pollard & Ivan A. Sag (1987) *An Information-based Syntax and Semantics*. Stanford, CA: CSLI.

[Raatz 1990] Stan Raatz (1990) *Graph-Based Proof Procedures for Horn Clauses*. Boston: Birkhaser.

[Robinson 1965] John Alan Robinson A machine-oriented logic based on the resolution principle. *J. Assoc. Comput. Mach. 12*, 23-41.

[Robinson 1979] John Alan Robinson *Logic: Form and Function: The mechanization of deductive reasoning*. New York: North Holland.

[Rodman 1977] Robert Rodman (1977) Axiomatizing Linguistic Structures. *Theoretical Linguistics 4*, pp. 255-264.

[Schwartz 1980] Myrna F. Schwartz, Eleanor M. Saffran & Oscar S. M. Marin (1980) The word order problem in agrammatism: I. Comprehension. *Brain & Lg. 10*, pp. 249-262.

[Schwartz 1985] Myrna F. Schwartz, Marcia C. Linebarger & Eleanor M. Saffran (1985) The status of the syntactic deficit theory of agrammatism. In M.-L. Kean (Ed.) *Agrammatism*. New York: Academic Press.

[Shankweiler 1989] Donald Shankweiler, Stephen Crain, Paul Gorrell & Betty Tuller (1989) Reception of language in Brocas Aphasia. *Language & Cognitive Processes 4*, pp. 1-33.

[Speas 1991] Margaret Speas (1991) *Phrase Structure in Natural Language*. Linguistic Inquiry Monographs ??. Cambridge, MA: MIT Press.

[Speas 1985] Margaret Speas & Naoki Fukui (1986) Specifiers and projections. *MIT Working Papers 8*. Cambridge, MA: MIT.

[Sportiche 1990] Dominique Sportiche (1990) *Movement, Agreement & Case*. Unpublished UCLA Ms.

[Stabler 1992] Edward P. Stabler Jr. (1992) *The Logical Approach to Syntax: Foundations, Specifications and Implementations of Theories of Government and Binding*. Cambridge, MA: MIT Press.

[Stowell 1981] Tim Stowell (1981) *Origins of Phrase Structure*. MIT Ph.D. dissertation.

[Travis 1984] Lisa Travis (1984) *Parameters and Effects of Word Order Variation*. MIT Ph.D. dissertation.

[Wehrli 1987] Eric Wehrli (1987) Parsing with a GB-grammar. In U. Reyle & C. Rohrer (Eds.) *Natural Language Parsing and Linguistic Theories*. Dordrecht: Reidel.

[Weinberg 1988] Amy Weinberg (1988) *Locality Principles in Syntax and in Parsing*. MIT PhD. thesis.

[Wu 1989] Andi Wu (1989) *"Jump in Difficulty" and the "Two-Tree Hypothesis" in Parsing*. UCLA M.A. Thesis.

[Zurif 1978] Edgar B. Zurif & Sheila E. Blumstein (1978) Language and the brain. In M. Halle, J. Bresnan and G. A. Miller (Eds.) *Linguistic Theory and Psychological Reality*. Cambridge, MA: MIT Press.

[Zurif 1983] Edgar B. Zurif & Yosef Grodzinsky (1983) Sensitivity to grammatical structure in agrammatic aphasics: A reply to Linebarger, Schwartz and Saffran. *Cognition 15*, pp. 207-213.