LOCALITY PRINCIPLES AND THE ACQUISITION OF SYNTACTIC KNOWLEDGE

by (Vol. I)

Robert Cregar Berwick

B.A., Harvard College
(1976)

M.S., Massachusetts Institute of Technology
(1980)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Doctor Of Philosophy
in
Electrical Engineering and Computer Science
at the
Massachusetts Institute of Technology

June, 1982

© Massachusetts Institute of Technology, 1982

Signature of Author Robert Cregar Berwick

Dept. of Electrical Engineering and Computer Science
April 27, 1982

Certified by

Noam Chomsky
Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Archives

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

OCT 20 1982

LIBRARIES

Locality Principles and the
Acquisition of Syntactic Knowledge

by

ROBERT CREGAR BERWICK

Submitted to the Department of Electrical Engineering and Computer Science
on April 27, 1982, in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in Electrical Engineering and Computer Science.

ABSTRACT

A principal goal of modern linguistics is to account for the apparently rapid and uniform acquisition of knowledge of syntax, given the relatively impoverished input that serves as the basis for the "induction" of that knowledge. This thesis presents a computational theory of the acquisition of syntax. It describes an implemented Lisp program that can acquire the rules to parse a substantial portion of English syntax, given only simple positive example sentences of English. In this research, an existing parser for English, Marcus' Parsifal, serves as the output representation of knowledge of syntax. We mimic the acquisition process by fixing a stripped-down version of Parsifal, thereby assuming an initial set of abilities. Simple pattern-action grammar rules are acquired, on the basis of examining grammatical example sentences with a degree of embedding of two or less.

From an engineering standpoint, the program can be viewed as an adaptable parser. Starting without any grammar rules, the currently implemented program has been able to acquire the majority of rules in a core grammar of English originally hand-coded by Marcus. It has also been able to acquire the rules to parse simple English dialect variants. From a theoretical standpoint, the acquisition program shows how the constraints of current linguistic theories can be embedded in a natural model of language acquisition. A developmental model of language acquisition is presented that formalizes this idea. Finally, the constraints on the operation of the acquisition program can be stated as specific locality principles that aid both parsing and acquisition. These constraints are the computational analogues of locality restrictions proposed in several current theories of generative grammar.

Thesis Supervisor: Noam Chomsky
Title: Institute Professor, Department of Linguistics and Philosophy

ACKNOWLEDGEMENTS

In the course of my research on learning, it has been my pleasure to have learned from so many. I would like to thank all of my "teachers" here.

-- Noam Chomsky, for his love of knowledge and his belief in the power of thought (even when he knows that our capacities are limited).

-- Mike Brady, Patrick Winston, Peter Elias, the other members of my committee, for their endurance and patience with long footnotes;

-- My good friends, Brian Smith, Amy Weinberg, Norbert Hornstein, Marilyn Matz, Candy Sidner, and David Israel, for all the priceless things that make them my good friends;

-- My friends and officemates, the legal firm of (Ed) Barton, (Dan) Brotsky, and (Bob) Sjoberg;

-- Two members of the Artificial Intelligence Laboratory, Patrick Winston and Marvin Minsky, for their never-flagging belief in the study of learning, or those that want to study it.

-- Beth Levin, Mitch Marcus, Ray Perrault, Chuck Rich, Craig Thiersch, Kurt VanLehn, and Ken Wexler, for all those things that make a scientific community a real *communitas.*

-- My family, and in particular my mother and father, for instilling in me a love of science and a love to find out *why.*

To all these teachers, and especially to the memory of my father, I dedicate this thesis.

1. Computational Constraints and Language Acquisition

One of the important goals of modern linguistic theory is to explain how it is that children can acquire their first language. This is a significant research problem: on many accounts the "evidence" that children receive to learn language is quite impoverished, and reinforcement by adults haphazard; yet the process of first language acquisition seems relatively easy and strikingly uniform. Children with enormously disparate sensory environments -- normal children, deaf children of normal parents, deaf children of deaf parents, normal children of deaf parents, blind children -- all seem, at least initially, to learn the same parts of what linguists call a grammar (Newport, Gleitman, and Gleitman, [1977]). Such robust performance in the midst of raging environmental variation poses a severe challenge for any theory of language acquisition.

Modern linguistics has countered with a research strategy that reconciles this apparent paradox by characterizing so narrowly the class of possible human grammars that the language learner's burden is eased, perhaps trivialized. In Chomsky's metaphor, hypothesizable grammars should be "sufficiently scattered" from one another so that children can easily select the one correctly corresponding to the language of their caretakers (Chomsky [1965]). Restrictions aid the learner because they rule out countless faulty hypotheses about which possible grammar might cover the language at hand. For example, suppose there was but a single human grammar. Such a situation would be optimal from the standpoint of language acquisition: no matter how complex the grammar, it could in principle be built-in, and no "learning" required. More realistically, current theories of transformational grammar are usually motivated and evaluated against the demand of learnability. In these theories, the possible rules of grammar (and thus the possible human grammars) are restricted as much as possible to just a few actions plus universal constraints on their application. The business of linguistics for the past several years has been to uncover these universal principles from the "data" of grammaticality judgments, and so advance, indirectly, an explanation for language learnability.

The metaphor of a child searching through a restricted hypothesis space of grammars has proved to be an enormously fruitful one for modern linguistic theory, drawing us closer toward the goal of characterizing what is distinctively human about the complex behavior that is called "language". However, the study of generative grammar as developed over the past few decades has, for the most part, deliberately *abstracted away* from a consideration of actual language use or the actual time course of language acquisition. As Chomsky has often pointed out, this is a scientific idealization of the usual sort, on a par with the assumption of, say, frictionless planes in physics. That is, it is an assumption, known to be false, that nonetheless allows researchers to focus on questions central to the theory of grammar while ignoring questions of human processing mechanisms that are not essential to explanation.

By formulating the inquiry in this manner, the theory of grammar is designed to tell us only what

knowledge of language is, not how it is used or how it is acquired. Of course, as suggested in the preceding paragraph, understanding *what* can often take us most of the way -- perhaps all of the way -- to understanding *how*, just as knowing the data structures involved in an algorithm will often tell one all one needs to know about how the algorithm using those representations actually runs. So one is left with the question: just how would one incorporate a theory of generative grammar into an account of how knowledge of language is acquired?

The goal of this thesis is to investigate this question in an explicit *computational* setting. First, a computational model for the acquisition of knowledge of (English) syntax will be set forth (Chapter 2). This is an implemented, working Lisp program, based on the Marcus parser [1980], that is actually able to "project" a substantial number of the parsing rules for English syntax based on an initial knowledge that consists of, roughly, a theory of grammar and input data that consists solely of simple, well-formed sentences of English. The first accomplishment of this thesis, then, is an engineering one: the acquisition procedure provides graphic demonstration that one *can* naturally embed a transformational theory of grammar into a working model of language acquisition. A substantial part of this thesis is devoted to an account of this model, the assumptions behind its structure, and its actual success in the acquisition of syntactic knowledge from relatively simple, positive-only evidence.

A second and related aim of this thesis is to explore the inter-relationship between learnability and parsability. As mentioned above, one of the background assumptions of modern linguistic theory is that the theory of universal grammar (UG) can be interpreted as part of a theory of language acquisition. By restricting the class of available grammars (compatible with evidence assumed available to the child), we advance toward an explanation of how it is that children can "fix" a grammar for the language of their caretakers. This exclusive attention to the functional demand of learnability has been criticized by computationalists. Why, they ask, should one concentrate on learnability and ignore the possibly equally constricting demands of parsability or sentence production? Evidently, not only are natural languages learnable, but sentences of natural languages are easily parsable and producible -- or at least those sentences that are observed in natural discourse.[1] If these objections are to be taken seriously, then a complete characterization of what is a natural, as opposed to an unnatural, language must address the problem of functional demands beyond that of

---

1 As we shall see, the fact that the corpus of sentences actually observed in day to day discourse is a finite portion of some infinite possible set of sentences causes some problems for the *direct* application of a functional demand of parsability or producibility, because only the finite portion matters in the functional evaluation with respect to parsability. In contrast, the entire infinite set could figure in the demand of learnability. For example, suppose it turned out to be the case that the set of sentences of a natural language was not even recursively enumerable. Then there would have to be an infinite number of sentences in the language that could not be recognized by any algorithmic procedure, and hence, *a fortiori* could not be parsed by any algorithmic procedure. But the behavioral import of this possibility could be insignificant -- all that we would observe is that people fail to use or produce certain sentences because they are too complex -- precisely what is observed, in fact, in well-known cases such as center-embedded sentences.

learnability. An important question then is to determine just how the multiple functional demands of learnability, parsability, and producibility interact. Pre-theoretically, this interaction could take many forms. For example, it could turn out that the constraints obtained on the basis of learnability considerations -- the specification of UG, in the linguistic sense -- are simply different from those constraints sufficient (or necessary) to meet the demand of efficient parsability. In this case, learnability and parsability constraints would simply turn out to be incomparable. Or, it could be that a specification of UG completely "solves" the problem of efficient parsability, in the sense that the constraints on grammars required to make the natural languages learnable also suffice to make such languages efficiently parsable with respect to those grammars. If this were so, then we would expect that it would be difficult, if not impossible, to *derive* universals of linguistic theory from considerations of parsability, as attempted by Marcus [1980]. Rather, we would find sentence processing operating so as to automatically obey the constraints of UG, and since all naturally attainable grammars would automatically satisfy the efficient parsability properties required in Marcus' framework, we would not be able to tell that this was for reasons of efficient parsability.

Plainly, it is difficult to assess the trade-off between acquisition and parsability with precision in the absence of either (i) a formal model of the general trade-off between acquisition and recognition complexity or (ii) specifically formulated models of acquisition or parsing. This thesis attempts to fill both these needs, presenting both a general, formal model to evaluate acquisition and recognition complexity and specific formalizations of the acquisition procedure described in Chapter 2 and the Marcus parsing model.

The general, formal model for the complexity analysis of competing acquisition and parsability demands is based on the notion of *program size* complexity. Informally, the amount of information required to "fix" a grammar on the basis of external evidence is identified with the size of the shortest program needed to "write down" a grammar. [2] The term "program" here is deliberately ambiguous, denoting either a computer program for producing a grammar, given some input data, or a developmental "program," in an ontogenetic sense. Given this interpretation of acquisition, one can formalize the usual linguistic approach of assuming that there is some kind of evaluation metric (implicitly defined by a notational system) that equates "short grammars" with "simple grammars", and simple grammars with "easily acquired grammars".[3] On this analysis, the more information that

---

2. The use of program size measures in inductive inference was introduced, so far as I know, by Solomonoff [1964] and studied in a general context by Biermann [1972]; see also Biermann and Feldman [1972].

3. This model identifies a notational system with some partial recursive function Φ_i(a Turing machine program), and a rule system as a program, p for generating an observed surface set of data (where the surface set of data is either a set of surface strings or a grammar itself, in which case the program p is a meta-grammar).

must be supplied to fix a rule system, the more marked or more complicated that rule system is.[4]

The program metaphor also provides some insight into the analysis of the ontogenesis of grammar acquisition, that is, how a grammar is acquired over time. Like any computer program, a program for a rule system will have a definite *control flow*, corresponding roughly to an augmented flowchart that describes the implicational structure of the program. The flow diagram specifies, among other things, a series of "decision points" that actually carry out the job of building the rule system to output. For example, one of the choices involved in fixing the base phrase structure system of a natural language is apparently whether Noun Phrase Objects (more precisely, Complement structure) appear to the left or to the right of Verbs. In English, an SVO language, NP Objects appear to the right of the Verb; in other, so-called SOV languages, as is the case apparently in Japanese, NP Objects appear to the left of the Verb. The selection of Head-Complement or Complement-Head order is presumably made on the basis of positive evidence available to the child; observational data indicates that this choice is indeed made early, and without apparent error. But once this order is set for the expansion of Verb Phrases, by the restrictions posited in X-bar theory, this order applies across all categories -- Prepositional Phrases, Noun Phrases, and Adjective Phrases. For example, Japanese has post-positional structure for all categories; English has pre-positional structure. Evidently, in this case the implicational structure of the developmental program is such that one bit of information, Head-Complement order, gives rise to a whole set of developmental implications, namely, the fixing of Head-Complement order across all categories. Thus a cluster of effects results from just a single decision. Put another way, natural (context-free) base phrase structure systems are more *regular* than the restrictions placed on them simply by the limits of the context-free re-writing formalism. If there is a rule of the form NP-->N NP PP S in a base rule system, then there will also be -- predictably -- a rule of the form VP--> V NP PP S. But no rules of the form VP-->N PP or VP-->NP V S PP will be found. This redundancy is what allows a significant compression in the "program" whose job it is to acquire a grammar, since a single decision serves to set the order NP--PP-S across all expansions of type XP, where X = N(oun), V(erb), A(djective), or P(reposition).[5]

In general, clusters of variables set in this fashion represent information theoretic redundancies, since the value of a whole group of parameters -- Head-Complement order for Adjectives, Prepositions, and Nouns -- is dependent on the value of another variable, namely, Head-Complement order in Verb

---

4. This account thus identifies *simplest* with *requiring the least additional information for specification* -- a definition that agrees with Sober's [1975] theory of simplicity, where *simplest* = minimum extra information.

The information-theoretic model presented above also subsumes several recent proposals concerning the role of evaluation metrics in linguistic theory: see Williams [1981b] and Chapter 3 for further discussion.

5. Recently, other authors have advanced roughly the same "developmental program" approach; see Roeper [1981]; Williams [1981c].

Phrases. In this case then, implicational structure in a developmental model corresponds rather directly to the existence of implicational clusters in the theory of grammar, regularities that admit short descriptions. In Chapter 3, we shall see that this same property holds more generally, in that all linguistic generalizations can be interpreted as implying specific developmental "programs" as described above. In particular, the way in which phonological systems are acquired will be shown to contain substantial information-theoretic redundancies of this kind. By interpreting these redundancies as a developmental program, one can actually explain the observed distribution of possible natural phonological systems as a side-effect of a simple model of acquisition from positive-only evidence.

In sum then, by construing the information supplied to a program as consisting of a specific sequence of data inputs, one can use the program metaphor as a way to reconcile the tension between *instantaneous* and *non-instantaneous* models of acquisition. An instantaneous model assumes that acquisition can be properly idealized as if all the input data were presented all at once to the acquisition device -- in other words, that there are no important acquisition constraints that arise out of the *ordering* or the sequence in which input data are presented to the acquisition device. Such models have been called instantaneous. This idealization has been the subject of much controversy. It has quite frequently, but informally, been claimed that developmental sequencing has an important role to play in the acquisition of grammar. The intuition behind this claim is that breaking down acquisition into a series of stages makes identifying grammars easier; at each step, there are fewer possibilities to consider, and new hypotheses can be built on the groundwork laid by past choices. This concern also surfaces in the claim of psychologists who suggest that the data input to children is richly structured, in the sense that mothers' speech to children is simple, highly inflected, repetitious, and so forth -- an almost Berlitz-like "training sequence". Part of the aim of this research has been to investigate this question. It will be demonstrated (Chapter 3) that there *are* cases where a non-instantaneous model give a better account of why a natural rule system is the way it is rather than some other way.

Just how does developmental sequencing matter in the acquisition of grammars? Formal analysis and experience with the acquisition procedure itself has led to the formulation of a set of specific *ordering criteria*, part of the temporal structure of the "program" that fixes a grammar on the basis of positive-only evidence:

> -- Rules that fix base phrase structure order are acquired before rules that fix local movement rules; in turn, rules that fix local movement are acquired before rules that fix general movement of constituents.
>
> -- Rules cannot contain arbitrary Boolean predicates on their triggering conditions.

-- If there is a distinction between obligatory and optional rules, then rules are assumed obligatory until explicit evidence indicates otherwise.[6]

-- Arguments to verbs are assumed obligatory until proven otherwise.

-- Arguments are assumed to appear immediately adjacent to Verbs (more generally, their operators) until proven otherwise.

On reflection it appears that all of these constraints arise from a single source: the demand that acquisition proceed on the basis of positive-only evidence. In fact, each of these constraints turns out to reflect a necessary and sufficient condition for acquisition from positive-only evidence. We call this condition the *Subset Property*. The Subset Property is a criterion to guarantee that a family of languages will be identifiable in the limit (in the sense of Gold [1967]) from positive-only examples. It states that identification is guaranteed just in case for all target languages L_i of a family of recursive languages $\mathcal{L}$, there exists an (effective) procedure that can enumerate positive samples S_1, S_2,... such that (i) $S_i \subseteq L_i$; and (ii) For all $j>1$, if $S_i \subseteq L_j$, then L_j is not a proper subset of L_i.[7]

The difficulty of acquisition given positive-only evidence also interacts with the theory of program size complexity and the program model of acquisition. In particular, it is shown in Chapter 3 that rule systems that are *disjunctively complete* (in a sense to be made precise) are difficult, sometimes impossible, to acquire from positive-only evidence. This result serves as a strong constraint on possible notational systems for grammars; for example, it means that one cannot admit notational devices that can be interpreted as disjunctive into the statement of rules, such as the use of braces in phonological or phrase structure systems. As a case in point, the constraint against disjunction implies that phrase structure rules of the following form are to be avoided:

VP--> V {NP or S-bar} e.g., VP--> V NP; VP--> V S-bar

VP--> V {NP or VP-bar} e.g., VP--> V NP; VP--> V VP-bar

Interestingly enough, the elimination of such disjunctions in phrase structure has recently received independent support in the work of Stowell [1981]. What the formal analysis shows is that these linguistic proposals can be independently justified on grounds of acquisition. As case studies of the interaction between the information-theoretic developmental model and acquisition constraints, three separate rule systems are analyzed: Kean's [1974] theory of markedness in phonological rule systems; the X-bar system for English; and the system of English AUX (or Inflectional system) (Chapters 2, 3).

---

6. Also proposed by Roeper [1981].

7. This condition has been implicit in the linguistic literature for several years (see, e.g., Baker [1979]), and was also formulated, apparently independently, by Angluin [1978] in a more formal recursive-theoretic setting. The condition as formulated above is taken from Angluin [1978]. See below, Chapter 3 for additional discussion.

To furnish specific models of the acquisition-parsing trade-off, this thesis formalizes both the Marcus model of parsing and the model of acquisition presented in Chapter 2 of this thesis. The "operating principles" of the acquisition procedure can then be compared to the most carefully articulated existing mathematical model for the acquisition of a generative (transformational) grammar, that of Wexler and Culicover [1980].[8] This comparison shows that similar constraining principles figure in the success of both the computational model of acquisition set out in this thesis and the Wexler and Culicover formalization. The confluence of constraint is not coincidental; as will be shown, both models incorporate essentially the same constraints on permissible grammatical rules. For Wexler and Culicover, the key learnability criterion that must be met for the languages generated by a family of grammars to be learnable from "simple" positive evidence is that the grammars must exhibit what they call "bounded degree of error" (BDE). Informally, the BDE property is a claim about language "separability"; it asserts that a sufficient condition for the learnability of a family of transformationally generated languages is that they be separable by simple data. Recall that a set of transformational rules (circa 1965) may be regarded as a function that maps a set of context-free generated *base* structures to *surface strings*. Briefly stated, the BDE property then simply says that a sufficient condition for learnability from simple data is that there exists a single bound on the depth of data, *d*, such that for every pair of (distinct) rule systems, T_1, T_2, and for every base structure *b* such that $s_1 = T_1(b) \neq T_2(b) = s_2$, there exists some other base structure *b'* of depth $\leq d$ such that $s_1^* = T_1(b') \neq T_2(b') = s_2^*$.

In Chapter 5, the BDE property, a sufficiency condition for learnability, is connected to parsability by showing that an extension of the *LR(k)* condition (Knuth, [1965]) subsumes the BDE criterion. This demonstration includes an automata-theoretic formalization of (an extended) class of Marcus-type parsers (The analysis shows that the class of extended-Marcus parsable languages and Degree-2 learnable languages includes some strictly context-sensitive languages.) Moreover, the formalization demonstrates that Marcus parsability implies Degree-2 learnability, in the sense that a family of languages with grammars that are each Marcus parsable is also Degree-2 learnable.[9] Thus, given these particular formulations of "efficient parsability" and "learnability", the demand of parsability actually turns out to be *stronger* than the demand of learnability. However, this finding must be interpreted with care, since the locality principles guaranteeing Degree-2 acquisition overlap so closely with those that ensure efficient parsing in the Marcus sense. Just for this reason it may be difficult and probably

---

8. The Wexler and Culicover work was also done with H. Hamburger (Hamburger and Wexler, [1975]), but, for convenience, this joint research will be referred to simply as "Degree-2 theory," because the acquisition model it posits uses only input sentences of degree of embedding 2 or less.

9. In this case "efficiently parsable" is defined as "executes in time linear in the length of the *derivation* of sentences". As will be shown, this criterion admits many of the simple examples of strictly context-sensitive languages discussed in the linguistics literature -- *xx* languages and $a^n b^n c^n$.

illegitimate to conclude that the functional demands of parsability *alone* or learnability *alone* have shaped the form of grammar that we have observed.[10]

In short, this thesis is an attempt to carry out quite literally the research program of generative grammar: to start with a characterization of *what* constitutes knowledge of language, and then go on to characterize *how* that knowledge is acquired. In a computational setting, this means a careful study of those aspects of acquisition that are essentially computational in nature, primarily those that have to do with the *order* of events, in particular a consideration of whether the order of sentences presented as input data has any bearing on the course of acquisition itself.

This "division of labor" into *what* and *how* has sometimes been taken to mean that one cannot properly proceed with a (computational) theory of acquisition of grammar without first settling upon, in detail, the "right" theory of grammar. Such an interpretation may well be too strong. It is often possible to take an non-computational theory whose details are not fully worked out (such as a theory of generative grammar), and embed that theory in a computational framework in order to gain additional insight into the representations implied by that theory. In the best case, such a move even allows one to admit additional sources of evidence, for example, assessments of processing time complexity, or other psychophysical measurements, to guide the construction of a better theory.[11]

One might compare in this regard a similar research strategy, that adopted by D. Marr and his

---

10. Furthermore, for several reasons the Wexler-Culicover acquisition model is probably not the appropriate framework in which to study the actual course of acquisition, particularly given current theories of generative grammar. For one thing, it seems to be too weak a theory. It is based on a weak model of acquisition, namely, one in which the learner discovers and corrects numerous errors in the formulation of transformational rules. This seems to be manifestly not the case for human language acquisition. Thus Degree-2 theory is probably best viewed as a kind of "upper bound" on what a linguistic theory must provide in the way of learnability. Given the status of Degree-2 theory, the fact that the class of languages parsable in the Marcus sense is narrower than the class of Degree-2 learnable languages is not so surprising. Because of these possibilities, even if turned out that the class of parsable languages was narrower than the class of learnable languages one still might not be able to conclude that parsability demands are more powerful than learnability demands. For example, the theory of UG might be narrow enough to ensure Degree-2 learnability. Parsability perhaps might then be obtained simply by adding the Marcus constraints of determinism and constituent look-ahead parsing – i.e., by "projecting" the UG class of attainable grammars through a small number of additional parsability demands. If this is correct, then the brunt of the observed constraints are borne by UG, rather than by parsability per se.

Finally, the notion of "efficient parsability" as typically used in the computational linguistics literature is suspect as a useful criterion when talking about specific functional efficiency claims. For a more thorough discussion of some of the issues surrounding so-called "functional" accounts of language, see Berwick and Weinberg [1981a]. The general conclusion supported there is that most *functional* arguments that have appeared in the literature founder on one or another (usually implicit) assumption about the actual, biological demands of language processing, the functional supremacy of parsing vs. acquisition, or the computational machinery assumed to be available.

11. This is not to say, of course, that this approach is necessarily *easy* one or that it will always succeed; in fact, as discussed in Berwick and Weinberg [1981b], there have been numerous *unsuccessful* attempts to use psychophysical evidence to bear on the choice among alternative theories of grammar, attempts vitiated, for the most part, because of a failure to consider alternative computational instantiations.

colleagues in the study of early visual processing. As Marr notes, in any research effort that is attempting to explain a complicated information processing system -- such as, one might suppose, the hypothetical "language faculty" -- explanations may be pitched at any one of several different *levels of description*:

> ...in a system that solves an information processing problem, we may distinguish four important levels of description... At the lowest, there is basic component and circuit analysis -- how do transistors (or neurons), diodes (or synapses) work? The second level is the study of particular mechanisms: adders, multipliers, and memories, these being assemblies made from basic components. The third level is that of the algorithm, the scheme for a computation; and the top level contains the theory of the computation. ... [T]ake the case of Fourier analysis. Here the computational theory of the Fourier transform -- the decomposition of an arbitrary mathematical curve into a sum of sine waves of differing frequencies -- is well understood, and is expressed independently of the particular way in which it might be computed.
>
> Marr and Nishihara [1978]; from Marr and Poggio [1977]

In Marr's view, one may distinguish the study of what is dubbed the *abstract theory of a computation* from the study of *particular algorithms* that realize that computation. This is the analogue of the distinction observed in the study of grammar. A theory of grammar corresponds to Marr's abstract theory of a computation: it tells us only *what* knowledge an ideal speaker-hearer has of language, abstracted away from any particular procedure that *uses* that knowledge (e.g., procedures for understanding or uttering sentences). And just as the abstract theory of a computation -- for example, the mathematical theory of the Fourier transform -- makes no reference to actual computational processes (indeed, makes no reference to *time* at all), the theory of generative grammar (almost by definition) does not involve the notion of *time*, and hence actual language use, at all. The theory of grammar is to be "expressed independently of the particular way in which it might be computed."

Given that Marr's and the generative grammarian's research strategies are so much alike, it is perhaps surprising that considerable dissatisfaction has been voiced with the grammarian's stance of abstraction, and much less with Marr's. It is hard to distill this uneasiness into a single line or two, but at bottom, it seems to amount to this: these critics of generative grammar (or even of particular theories of grammar) sometimes seem to be saying that a theory that begins and *ends* with grammar cannot be a true account of the human language faculty. Marr, it should be noted, does not stop in his account of early visual processing at the abstract theory of the computation; he goes on to probe alternative algorithmic realizations of that theory, and a range of possible machine implementations for those algorithms, ultimately aiming for a full account of the *psychophysical* behavior of human visual perception. Isn't it then possible, these critics go on to say, that by *ignoring* the exigencies of computation -- associated algorithms and machine implementations for grammars -- that one could arrive at a theory of grammar that, loosely speaking now, literally could not be incorporated into the

human nervous system, i.e., a theory of grammar that could not be "psychologically real"? Note that, in Marr's terms, this would mean that we would have arrived at a theory of the abstract computation for which there was *no* associated algorithm.

Returning now to the aims of this thesis, another way to view the research reported on here is as an *explicit* demonstration that these fears of "non-realizability" are unfounded. The acquisition model does the double-duty of (1) accounting for language processing via a system closely tailored after the structures provided by generative grammar; and (2) accounting for the unfolding of those processing abilities over time. Indeed, if the connections between Wexler-Culicover learnability and parsability are correct, then not only can generative grammars for natural language be "realized" in models of parsing or acquisition, they can be *efficiently* realized. In short, it appears that one can adapt Marr's levels-of-representation framework to the study of language, and explore alternative algorithmic realizations of theories of grammar as models of sentence processing, acquisition, or sentence production.[12]

Adopting the general research stance taken in the study of the visual processing system then, the first task is to supply a process model of language use that is grounded upon the rules and representations of a generative theory while at the same time faithfully mirroring "natural" parsing behavior. One such model already exists: Marcus [1977, 1980] has developed an efficient, left-to-right parser for English (Parsifal) that closely mimics the "operating principles" of Extended Standard

---

12. Some have questioned just why a single process model should be made to serve both as a model of language use and as the cornerstone of an acquisition model. For instance, Fodor, Bever and Garrett [1974] suggested that substantially different mechanisms ("rule systems") are involved in language acquisition and in language use. One problem with this suggestion is that if language acquisition and language use are completely disassociated, then it would be difficult to explain how the mechanisms engaged in language use are acquired or why knowledge of language should be acquired in a form that is not used. Without adopting this radical proposal, one can formulate other, intermediate models that distinguish between acquisition and use. Fodor, Bever, and Garrett proposed that the representations (roughly, data structures) might be the same for acquisition and language use, though the actual algorithms operating with those data structures in parsing or production could differ, perhaps widely. The research reported on here has adopted a variant of this position, namely, that the actual representations acquired can be embedded rather directly in a model of language use, in this case, a parser.

An additional question that can be raised about the approach taken in this research is its concern with *parsing* rather than *production*. Since in a broad sense the job of a parser is to map between input strings and a formal (mental) representation that is to be interpreted, and since sometimes ill-formed syntactic strings are input that nonetheless require interpretation, it has sometimes been argued that a parser must be built so as to err on the side of over-acceptance – i.e., it must accept a wider class of sentences than just the grammatical ones. Conversely, it has been claimed that a production device ought to be designed so as to err on the conservative side, never producing sentences that might be ill-formed. The possible symbiosis between parsing and production has not been investigated in this work.

transformational theory.[13]

In the classical generative account, "syntactic knowledge" is characterized simply as a way to generate a list that pairs *surface strings* (sentences) with *structural descriptions* of those strings (labelled bracketings, or parse trees). Hence the term "generative grammar." A transducer (such as Parsifal) adds an inherent directionality to this account by providing an effective procedure for passing from any given surface string to its proper labelled bracketing (if the string is syntactically well-formed) or some error state (if it is not).[14]

The Parsifal procedure itself can be thought of as a simple two-part machine: an interpreter (a finite-state control, a fixed collection of data structures and a working storage that might or might not be infinite in extent) and the grammar rules (machine instructions) that the interpreter executes. It is the grammar rules that do the actual work of the parse, unwinding the mapping between surface form and structural description; the interpreter serves as bookkeeper and control. The transducer's storehouse of rules is thus a repository of syntactic knowledge *as it functions in language use*, in this case, parsing.

With Parsifal in hand, the next step is to model the acquisition of syntactic knowledge as the acquisition of a series of parsers of increasing sophistication. This is the focus of the research described in this report; the approach is sketched out in Figure 1.1. below. First, one fixes some initial state of knowledge, corresponding to an inital set of parsing abilities and a knowledge of what counts as a valid rule of parsing. This state is labelled as "$P_{initial}$" in Figure 1.1. Then, an acquisition procedure constructs a sequence of new transducers, P_i, incrementally adding to or modifying the

---

13. Chapter 2 offers some modifications to Parsifal that would bring it more into line with current theories of transformational grammar as discussed in Chomsky [1981].

More precisely, Parsifal is a *transducer*: it takes as input input sentences and, as it processes sentences, it produces as output a parse tree. For convenience, this thesis will often interchange the terms *parser* and *transducer*. Parsifal adheres to the representational format of transformational theory circa the mid-1970's in that its output trees that are, to a first approximation, annotated surface structures as described, in e.g., Chomsky [1975]. Marcus [1980] also argued that Parsifal incorporated the operating principles of this theory in that it appeared to obey, for quite independent reasons, such constraints as Subjacency, the Specified Subject constraint, and some versions of the Complex Noun Phrase constraint. For further discussion, see Chapter 5 of this thesis.

From this confluence of operation it is, of course, not legitimate to immediately conclude that Parsifal embodies a transformational grammar in the same sense that, say, a (non-deterministic) push-down automaton "embodies" a context-free grammar. This conclusion would be warranted if and only if one could show that (i) for every transformational grammar, one could write a set of Parsifal grammar rules to parse all and only the sentences generated by that grammar (up to "transparent" truncation effects imposed by limitations of finite memory, analogous to the way in which a finite-stack push-down automaton mimics an associated context-free grammar up to some limiting depth of center embedding) and (ii) for every Parsifal transducer, there exists an associated transformational grammar (of a mid 1970's form).

14. The error state need not be a simple "dead state", but could range anywhere from an opaque "error" message to an incomplete parse tree (up to the point of failure) along with suggestions as to possible reasons for the failure of the parse.

knowledge base of parsing rules in response to a set of input data. At each step one is guaranteed the existence of a machine P_i that can process a subset of the language at hand in accordance with the more abstract rules of grammar uncovered by linguistic theory; yet at the same time, the sequence of transducers mimics an (idealized) acquisition process by utilizing these same constraints to approximate ever more closely a "mature" knowledge of syntax.

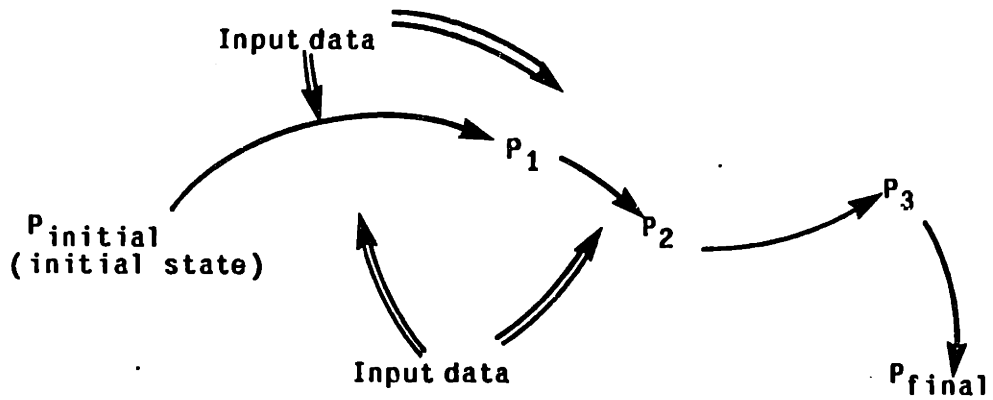


Figure 1.1 - Acquisition as modelled by a sequence of parsers.

This model owes much to the picture of acquisition outlined in Chomsky's *Aspects of A Theory of Syntax*:

> A child who is capable of language learning must have
> (i) a technique for representing input signals
> (ii) a way of representing structural information about these signals
> (iii) some initial delimitation of a class of possible hypotheses about language structure
> (iv) a method for determining what each hypothesis implies with respect to each structure
> (v) a method for selecting one of the (presumably, infinitely many) hypotheses that are allowed by (iii) and are compatible with the given primary linguistic data
> [1965, page 30]

> This [language acquisition] device must search through the set of possible hypotheses G_1, G_2, ..., which are available to it by virtue of condition (iii), and must select grammars that are compatible with the primary linguistic data, represented in terms of (i) and (ii). It is possible to test compatibility by virtue of the fact that the device meets condition (iv). The device would then select one of these potential grammars by the evaluation measure guaranteed by (v).
> [1965, page 32]

It is easy to see what the current proposal and the *Aspects* model have in common. Both presume an ability to represent the speech signal as a highly structured set of input tokens -- in the case of the acquisition procedure, as the segmented individual words input to the parser. (For further discussion of the developmental assumptions behind this proposal, see Chapter Two, Section 2.) Both propose to represent those signals as highly abstract, labelled tree structures. Finally, both approaches presume an initial delimitation of the class of possible hypotheses about those structures -- in the case of the *Aspects* model, initial constraints on grammars, and for the acquisition procedure, a pre-supposed initial parser (perhaps just a bare interpreter), universal constraints on the parser's operating principles, and a knowledge of what counts as a valid rule of parsing.

It is true, however, that the *Aspects* model has deliberately (and even legitimately) abstracted away from the actual time course of acquisition in its attempt to arrive at a perspicuous idealization. In a word, it is an instantaneous model of acquisition: it lumps all possible candidate grammars and data together, and, using a selection function (the evaluation measure, or evaluation metric) picks out the "best" grammar (the "correct" G_i) at one fell swoop. As Chomsky notes in the footnote to the passage quoted immediately above, this is once a again a scientific idealization of the usual sort, an assumption known to be false, yet apparently innocuous:

> What I am describing is an idealization in which only the moment of acquisition of the correct grammar is considered. ... it might very well be true that a series of successively more detailed and highly structured schemata (corresponding to maturational stages, but perhaps in part themselves determined in form by earlier steps of language acquisition) are applied to the data at successive stages of language acquisition.
> [1965, footnote 19 to Chapter One, page 202]

The research reported on in this thesis attempts to relax this idealization of instantaneous acquisition, and pursues the consequences of adopting an explicit developmental approach. There are at least two possible rewards to this approach. First of all, a problem for any "learning theory" is to account for how a learner is "driven" from one stage of knowledge to the next:

> There is, however, a very general problem with practically all studies of language development, whether investigated from the standpoint of rule acquisition, strategy change, or elaboration of mechanism. The problem arises both for accounts that postulate 'stages' of development (i.e. a finite number of qualitatively distinct levels of organization

> through which the organism passes en route from molecule to maturity), and for accounts that view development as a continuous function of simple accumulation. The difficulty is this: No one has seriously attempted to specify a mechanism that 'drives' language acquisition through its 'stages' or along its continuous function. Or more succinctly: there is no known learning theory for language.
> [Marshall, 1979 page 443]

The current model is designed to remedy at least part of this problem: it is the parser's attempts to interpret sentences that provides a specific driving mechanism for acquisition. This model constitutes a true theory of acquisition.

Secondly, even though a developmental approach need not necessarily provide any new insights into the structure of grammar, it could be useful in its own right as a theory of the actual time course of acquisition. In other words, if one was interested in explaining what actually *happens* in acquisition, then a theory that attends to the time course of events would seem to be crucial. Such a theory is of obvious interest to those who are interested in developmental processes in and of themselves. Thus, those who are interested in why what looks like a rule of "truncated passives" is acquired earlier than a rule for full passives, or why auxiliary verb inversion appears to be acquired at different times depending on which *wh*-question word is involved, must look to an essentially developmental theory, perhaps of the kind described here. Note that there is no necessary contradiction between the instantaneous and non-instantaneous viewpoints, but simply a difference in research aims. Chapter 2 discusses some of this developmental evidence in passing.

Finally, but more speculatively, by incorporating the actual time course of events into an acquisition model we may hope to gain further insight into what distinguishes the class of natural languages from arbitrary symbol systems. As discussed above, the belief here is that there may actually be developmental dependencies that further restrict the space of hypothesizeable grammars -- dependencies such that earlier choices constrain the choices that must be made at a later stage. The advantages accruing from a developmental approach in this regard are more uncertain. Staging contingencies are, of course, quite common in the development of other biological structures, but whether such dependencies play a crucial role in the acquisition of grammars is open to question. This matter is taken up in Chapter 3.

What does it mean for the notion of *time* or *developmental stages* to play an essential role in an acquisition process? In the "search space" metaphor of acquisition, it amounts to the claim that grammars are separable (hence acquirable) in part as the result of the *order* in which certain decisions are made. Put another way, it asserts that once, say, part of the rule system for grammar G_i has been fixed, then certain other options, say those that are part of grammars G_j or G_k can never be acquired, or are inaccessible. Chapter 3 draws on the particular case of regularities in phonological systems (from [Kean 1974]) to show that this developmental account may in fact actually explain why certain

phonological systems do not occur naturally.

To say then that time plays an essential role in acquisition is to say, analogous to the embryological case, that once certain pathways have been selected (on the basis of specific triggering evidence), other options may be cut off forever (or at least severely restricted). To say that time plays an *essential* role is to make the stronger claim that restricting developmental pathways in this fashion is necessary for the successful identification of grammars for natural languages. For example, consider again knowledge of the "canonical" ordering of constituents in an English Noun Phrase. As several authors have noted,[15] the basic order of Article-Noun or Adjective-Noun -- such as in *a bottle* or *big dish* -- appears early in a child's productions. Furthermore, once these forms are established, gross errors such as **book a*, do not arise. One could account for this rapid fixation on the right phrase structure by postulating the existence of a structural "parameter" of all Noun Phrases that would permit, say, Articles to appear to either the right or the left side of the Noun, but not both sides. The decision as to right or left attachment could then be left to the actual evidence the child hears: upon encountering (innumerable) forms like *the doll*, one could conclude that the sequence "Article"-Noun is the "right" setting of the option, the sequence Noun-Article, not so allowed. This "parameterized" view of acquisition reflects directly the so-called "X-bar" theory of phrase structure (for further discussion, see

15. See, e.g., Williams [1981b].

Chapter 3).[16]

1.0.1 The acquisition model, Lparsifal

For the developmental approach to succeed we must spell out in full just how Chomsky's components (i)-(v) are realized in the new framework -- exactly what is meant by initial parser, input data, and acquisition procedure. That is the goal of this report: to document and justify the details of an implemented LISP program, Lparsifal, that is designed in accordance with Figure 1.1. As its initial parser, the Lparsifal procedure employs a streamlined, rule-less version of Parsifal -- roughly, the bare interpreter and its data structures. As input data, the program takes just grammatical sentences (so-called *positive evidence*) and a rudimentary initial ability to characterize words as objects, actions, or unknown. (The ability to label lexical items with category features has a developmental program of its own, interacting with unfolding syntactic abilities; for further discussion, see Chapter Two, Section 2.) The developing parser attempts to analyze each example sentence supplied by the external

---

16. It is interesting to observe that this model of the ontogenesis of grammars follows almost exactly that of embryological models. In the "classic" cases of embryological development we find that the notion of *time* does matter, in the sense that specific "triggers" are required at certain points in time for development to proceed on course. For example, in sea urchins the initial cleavage of the fertilized egg splits the single cell into two parts containing a differential distribution of cellular material -- the so-called "animal" and "vegetal" poles. This initial assymmetry provides the foundation for later differential development -- the new "vegetal" cell is destined to become cells quite different in function from those of the "animal" one. But how is the initial gradient of cellular material established? The answer is that the system exploits an absolutely regular and dependable environmental "trigger" -- gravity. Note that precisely because the genetic system can assume that gravity acts so as to provide an initial asymmetric distribution of material, a simple division along a horizontal plane divides the original fertilized egg into two differentiated new cells. This example exhibits all the properties of developmental triggers. First, the trigger is quite abstract. The sea urchin system does not "learn" that gravity causes a differential distribution of cellular material; rather, this environmental regularity is exploited in an indirect fashion. Second, almost the entire information content of the system is tied up in the genetic process, not in the trigger. In fact, the trigger may be thought of as exactly one "bit" of information -- whether gravity has yet induced a gradient or not. Third, even though the triggering information is minimal, it is *necessary* for the functioning of the system; indeed, the sea urchin's development depends on it. Just for this reason, the external trigger *must* be an absolutely dependable environmental regularity -- such as gravity.

Now observe that each of these properties of embryological development is met by the X-bar account of acquisition. Relative to the total information in the X-bar system (now assumed to include information about lexical items) the information content of the "trigger" that says that in English Specifiers appear to the left of Heads is quite low. Further, the triggers themselves are absolutely dependable external regularities: just as gravity is ubiquitous, so too are English sentences where Complements can be seen to follow Heads, and Specifiers precede Heads. Third, the trigger is a necessary part of the acquisition of the right grammar, even if it does not dominate the information theoretic content of grammatical development. Note that this possibility might account for some of the confusion surrounding the "unreality" of an instantaneous model of grammar acquisition. If it turns out that grammar acquisition is just like that of embryological development, then one could maintain an assumption of instantaneous acquisition and still obtain a nearly complete picture of the space of possible natural grammars, in an information-theoretic sense. Yet it could also turn out that a *complete* account of development, particularly with regard to predictions about the actual course of grammar acquisition, would demand a description of that small set of points where environmental regularities intervene.

world;[17] the acquisition of new rules is prompted by rule failures during the parsing process. A failure is defined simply as either (i) the failure of any known grammar rule to apply; or (ii) the application of a known rule that results in a base structure known to be incorrect.[18] Each time that a failure is detected on-the-fly in the left-to-right processing of a sentence and successfully resolved, the system adds a single new parsing rule to its knowledge base or modifies (perhaps reinforces) an old rule. If the system cannot resolve a local parsing failure, it simply gives up on the analysis of the current sentence and tries to parse the next. This design decision gives the system an incremental behavior; it tends to acquire its knowledge from "clear cases" rather than making large inferential steps. It is this characteristic that enables the system to mimic non-instantaneous development.

Three sorts of syntactic knowledge are acquired:

> (1) The ordering information implicit in Context-free phrase structure rules. The knowledge acquired here corresponds to the basic order and branching structure of (English) phrases, e.g., that a Verb Phrase consists of a Verb followed by a possibly optional Noun Phrase. (Note that no commitment is made to precisely *how* this ordering information is expressed; one need not use explicit phrase structure rules.)
>
> (2) Pattern-action lexical insertion grammar rules. This knowledge corresponds roughly to valid contexts for the insertion of terminal elements into a phrase structure tree. (This information might also be viewed as *local transformations*.) As a by-product, lexical categorizations are developed

---

17. The order of presentation may be considered random; that is, the external environment does not impose a structure on the order in which examples are given. Allowing a highly structured input of this kind can be shown formally to trivialize the task of acquisition, because one can encode the "right" hypotheses in the presentation order itself. (See Chapter Two, Section 2 for additional discussion, or Gold [1967].) A characterization of what, exactly, is the primary data that children receive to learn language has been a topic of vast dispute over the past several years (see discussion in Chapter Two and Appendix Two). The ordering of examples actually provided to children is probably neither completely random nor so structured as to provide possible encodings of grammars. Yet despite the immense possibility for individual variation in data input, the course of acquisition is remarkably uniform. This uniformity can only come from two sources: the external world or the child. In light of the controversy over what is input to the child, the current research has opted for a strong assumption about external order, namely, to assume no external ordering of input data; all structure is imposed by the internal structure of the acquisition model itself. Note that even with this assumption there is a device whereby the effects of randomness can be partially "filtered" so as to ensure that (by and large) simple sentences are acquired before more complex examples. One might call this the principle of *data focus*: the competence of the system at any given point in time imposes an intrinsic filtering of the input examples, in that examples "too complex" for the parser to either (1) handle directly or (2) easily acquire rules for (in a sense of "easily acquire" that will be made precise later) are simply ignored. The first sentence of *Finnegan's Wake* might be input first, but it will simply not be (completely and successfully) interpreted, and thus not figure prominently in acquisition. Again, see Chapter Two, Section 2 for additional discussion.

18. This second failure introduces the system's contact with a crude kind of semantic interpretation, in the sense that incorrect base structures can be detected in simple cases via the recovery of *thematic roles* such as Agent, Intrument, Theme, etc. independently of linguistic context. This assumption is comparable to that of Wexler and Culicover [1980], who assumed that base structures were somehow inferrable "from context."

on-the-fly.[19] For example, a rule stating that a transitive verb like *kiss* must appear with a Noun Phrase Object embodies contextual information of this kind.

(3) Pattern-action movement rules. This knowledge corresponds to particular movement operations, inversions, and the like, e.g., NP-movement, Subject-Auxiliary verb inversion in questions. (Such rules are the parser's counterpart of a general *move alpha* movement rule.)

This simple acquisition procedure has proved to be remarkably successful: the currently-implemented version of the model can acquire from just simple positive example sentences the parsing rules sufficient to handle a large core subset of English sentences. Syntactic phenomena as diverse as Subject-auxiliary verb inversion, simple passives and constituent movements, as well as the "canonical" ordering of phrases in an English sentence can all be acquired by the procedure. (See the Appendix to this Chapter, repeated in Chapter 2, for a complete list of the parsing rules the procedure can currently acquire.)

19. See Chomsky [1965] footnote 18 to Chapter 2 for a discussion of local transformations; See Joshi and Levy [1977] for a formalization of the notion of *local transformation*.

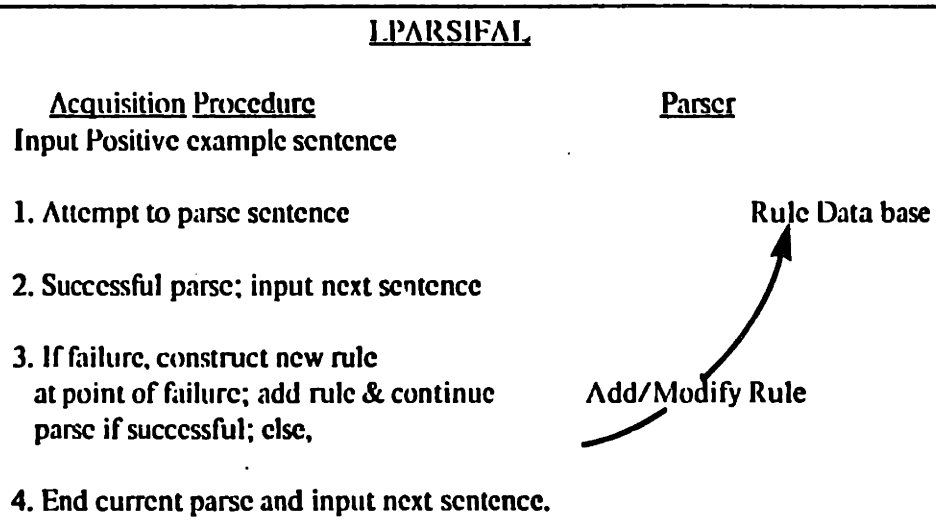


Figure 1.2 - Outline of the acquisition procedure.

Note that according to this model, new knowledge is acquired by modifying only the rule data base of the parser; the interpreter (working storage and control structure) and knowledge of the valid forms for rules remain fixed. [20] This division is assymetric. However, it appears to be a legitimate working assumption: the interpreter and the knowledge of constraints on rules are presumed to be universal across all languages, hence in fact fixed. There are, however, several details of the interpreter that might in fact change over time. One can imagine a model where, for example, the working storage of the interpreter is expanded over time (perhaps corresponding to a general increase in memory or attention span) or the inventory of rule features altered (again in response to the concurrent development of other cognitive faculties). But no extensive revisions in the control structure of the interpreter seem to be required, in part because the basic execution loop of the "mature" interpreter is

20. One might well question whether the model's identification of grammatical knowledge with a set of *rules* is the right approach. As Chomsky has noted, there has been a shift in recent years in generative grammar away from the study of rule systems and toward the study of systems of *principles*. In this newer view, rules are just the "surface" epiphenomena of deeper systems of constraints that underly grammatical systems. Thus, there is no "rule of passive" as such but rather a cluster of properties that arise out of the interaction of a single movement rule and a number of quite general constraints on grammars.

already so simple that there is really very little to learn.[21]

In summary, the proposed model for acquisition is quite simple. Knowledge of syntax (a grammar) is represented as a parser; development of that knowledge by a sequence of parsers. The acquisition process itself is driven by the current parser's attempts to interpret positive example sentences prompting changes to its data base of parsing rules.

21. There are *some* organizational modifications that are obviously worth investigating. For instance, the ability to handle the "interrupts" characteristic of the processing of recursive phrases (especially, given Marcus' framework, the parsing of Noun Phrases as separate entities), is a plausible candidate for maturational change.

1.1 Computational Analogues of Constraints on Grammars

1.1.1 Computational Constraints and Representational Constraints

The assumption that languages are structured in order that they may be easily acquired receives strong support from the results of this research. Simply put, what makes the rules easy to acquire is that the choices that must be made are few; the acquisition program is limited to constructing rules only of a certain kind, built from but a handful of possible actions. The success of this approach thus also confirms what is fast becoming a truism in artificial intelligence: having the right restrictions on a given representation can make learning simple.

What gives further support to this finding is the isolation of several key computational principles that are apparently responsible for the program's success, specific locality principles that tightly restrict the operation of the parser and the acquisition procedure. Central here are Marcus' key restrictions on the original Parsifal: (1) structural "determinism" (once the parser builds structure, its decisions cannot be undone); (2)left-to-right operation (the parse tree is built in a single left-to-right pass through the sentence) and (3) bounded context parsing (the parser makes its decisions based on the local context of the parse). In other words, the constraints that make acquisition easy can be classified into one of two groups: constraints on rule application -- the locality principles -- and constraints on rule form -- the pre-specified format for base phrase structure rules and grammar rules.

The way in which Parsifal imposes these constraints is straightforward. Take first the restrictions on rule application. Parsifal grammar rules consist of simple productions of the form

if ⟨*pattern*⟩ **then** ⟨*action*⟩

where a *pattern* is a set of feature predicates that must be true of the current environment of the parse in order for an *action* to be taken. *Actions* are the basic tree-building operations that construct the desired output. Adopting the operating principles of the original Parsifal, grammar rules can trigger only by successfully matching features of the (finite) local environment of the parse, an environment that includes a small, three-cell look-ahead buffer holding already-built constituents whose grammatical function is as yet undecided (e.g., a Noun Phrase that is not yet known to be the subject of a sentence) or single words. It is Marcus' claim that the addition of the look-ahead buffer enables Parsifal to always *correctly* decide what to do next -- at least for English. The parser uses the buffer to make discriminations that would otherwise appear to require backtracking. Marcus dubbed this "no backtracking" stipulation the *Determinism Hypothesis.* The Determinism Hypothesis crucially entails that all structure the parser builds is correct -- that already-executed grammar rules have performed properly. This fact provides the key to easy acquisition: if parsing runs into trouble, the difficulty can be pinpointed as the current locus of parsing, and not with any already-built structure (previously executed grammar rules). In brief, any errors are assumed to be *locally* and *immediately* detectable.

This constraint on error detectability appears to be a computational analogue of the restrictions on a transformational system advanced by Wexler and his colleagues. (Wexler and Culicover [1980]). In their independent but related formal mathematical modelling, they have proved that a finite error detectability restriction suffices to ensure the learnability of a transformational grammar, a fact that might be taken as independent support for the basic design of Lparsifal. Wexler and Culicover assure that their acquisition procedure will converge in finite time by keeping the number of hypothesized rules small, a situation guaranteed by restricting the context of rule application. Context restriction, in turn, is guaranteed through a host of stipulations, including what Wexler and Culicover call the Binary, Freezing, and Raising principles. Without going into extensive detail about the exact formulation of these principles, their rough effect is to limit the scope of rule actions to a small radius about a rule's point of action. A detailed point-by-point comparison of the Lparsifal and Wexler and Culicover proposals shows how close the two systems really are; see the figure below. Almost all the constraints assumed in the Wexler and Culicover work are mirrored in one form or another in the Lparsifal system. Chapter Five of this report explores the connections between them in a more systematic fashion.

Turning now to constraints on rule *form*, it is easy to see that any such constraints will aid acquisition directly, by cutting down the space of rules that can be hypothesized. To introduce the constraints, we simply restrict the set of possible rule <*patterns*> and <*actions*>. The trigger *patterns* for Parsifal rules consist of just the items in the look-ahead buffer and a local (at most two node) portion of the parse tree under construction -- five "cells" in all. Thus, *patterns* for acquired rules can be assumed to incorporate just five cells as well. As for *actions*, a major effort of this research has been to demonstrate that just three or so basic operations are sufficient to construct the annotated surface structure parse tree, thus eliminating many of the grammar rule actions in the original Parsifal. (It is these restrictions on rule patterns and actions that ensure that the set of rules available for hypothesis by the acquisition procedure is finite.)

1.1.1.1 Formalizing the Locality Principles

As mentioned earlier, one of the goals of this thesis is to provide an explicit formalization of the Marcus parsing model and its acquisition counterpart so as to be able to make precise comparisons between the Wexler-Culicover theory and the model advanced in this thesis.

Our first task is to formalize the Marcus parser. Marcus [1980] has claimed informally that his parser is really just an LR(k) parser, in the sense of Knuth [1965], but with its lookahead based on constituents (nonterminals) rather than just terminal items, as is conventional. That is, the lookahead buffer may hold entire consistuents, such as NP's, and grammar rules may trigger on the presence or absence of constituents. Interestingly enough, in his original article on LR(k) parsing, Knuth [1965] himself

suggested that lookahead could just as well be based on nonterminal as well as terminal elements. Thus the actual design of the Marcus parser is not novel in this respect. Further, an LR(k) parser obeys a strict recognition sequence that is not obeyed by the Marcus parser. Recall that an LR(k) parser analyzes an input sentence Left to right, producing a Rightmost derivation in reverse. Suppose for example that the grammar generated the NP "the boy John knows" via the following right-most derivation: NP --> NP S --> NP NP VP --> NP NP hit --> NP John hit --> Determiner Noun John hit --> Determiner boy John hit --> the boy John hit. Then recognition of this string would follow this derivation order in reverse. First the Determiner "hit" would be analyzed; the Noun "John"; then these two elements would be bundled together as an NP, and so on. But the Marcus parser violates this canonical derivation sequence: as soon as an unambiguous "leading edge" of a Noun Phrase is detected, in this case, the determiner "the", the parser predicts that an entire NP will be found. Thus the Marcus parser is non-canonical, interspersing top-down prediction of phrases with bottom-up assembly of phrases. For instance, in the parse of "Have the boys taken the exam yet?", the decision to reduce the token *have* back to either an auxiliary verb (as in this case) or a main verb (as in the imperative, "Have the boys take the exam!") is delayed until the entire NP following *have* is reduced. Thus a reverse right-most derivation sequence is not obeyed:

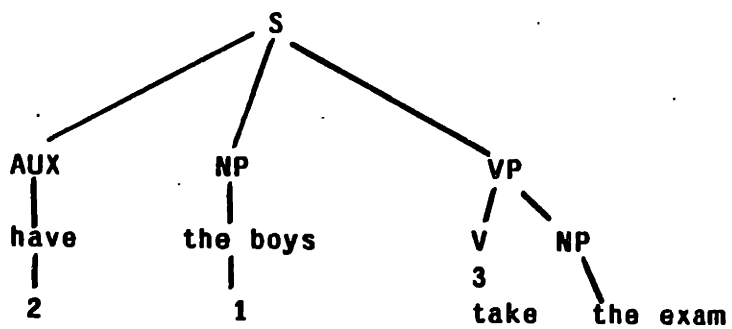


Figure 1.3 - Non-canonical recognition order.

The ability to handle non-canonical derivation sequences was implemented by Marcus by introducing a look-ahead buffer that could hold constituents as well as words, and a mechanism whereby the parser could be called, recursively, on this look-ahead buffer. The resulting parser is a non-canonical, two-stack, extended LR(k) parser:

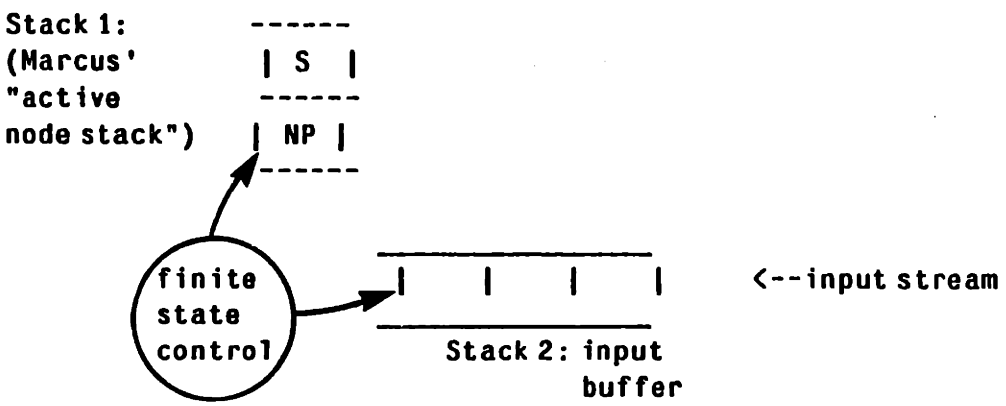


Figure 1.4 - The Marcus machine as a 2-stack device.

Note that Marcus' "left-to-right" constraint is replaced by the push-down stack behavior of the buffer.

Some other changes must be made to the Marcus machine so that it can be formalized. The machine is restricted in two ways so that we do not have an arbitrary, two-stack deterministic push-down automaton (and hence Turing-complete power). First, it is defined to be halting, so that indefinite cycling is not permitted. Second, it must possess a certain local error detectability property, just like an LR(k) machine. Finally, in order to turn the Marcus device into a more conventional form, one must eliminate the way that sub-trees are built into the left-hand stack. This information can be recoded as single, though complex, non-terminal symbols. Details are provided in Chapter 5.

Given this formalization, one can now examine more precisely certain intuitions about the Marcus parser. For example, if recursive calls to the parser ("attention shifts" in Marcus' terminology) can never exceed a fixed, finite bound, then Szymanski and Williams' [1976] results on non-canonical parsing can be used to show that the languages accepted by Marcus-type parsers must be deterministic. (Under most assumptions, the languages must be deterministic context-free languages, but see Chapter 5 for extensions that allow, e.g., $a^n b^n c^n$ to be parsed.) Chapter 5 discusses these matters in more detail.

Finally, as mentioned, Chapter 5 examines the connection between the Wexler-Culicover BDE condition and LR(k) parsability, as extended to a two-stack model. The close relationship between the independently developed Lparsifal and Wexler-Culicover models is summarized in the table below.

Culicover and Wexler Constraints	Acquisition Procedure Constraints
Incremental rule acquisition	Incremental rule acquisition
Universal base (can be weakened assuming a theory of base rule base rule acquisition)	Universal base (can be weakened assuming a theory of base rule acquisition)
NO negative external evidence	NO negative external evidence
Only current sentence used to construct new rule	Only current sentence used to construct new rule
Small number of new rules available for hypothesis	Small number of new rules available for hypothesis
Rule construction based on "simple" data: depth of embedding at most two	Rule construction based on "simple" data: depth of embedding at most two
Binary principle	
Freezing principle	Determinism plus locality restrictions imposed by buffer and active node stack
Raising principle	

Figure 1. 5 - Constraints advanced by Culicover and Wexler vs. those of the acquisition procedure.

Apparently then, the exigencies of language processing (parsing) interact are intimately connected to the abstract constraints advanced by current theories of grammar; many of the same constraints that ease parsing -- forcing decisions to be made locally -- also aid the cause of acquisition. In brief, paying deliberate attention to the constrained theories of grammar developed by linguists has paid off handsomely: on the one hand, it has permitted the construction of a working computer program for the acquisition of syntactic knowledge; on the other, it has confirmed the value of theories of grammar as the basis for reasonable processing models of language acquisition and language use.

1.2 Evaluating the Model

As Pinker [1979] has noted, any model of the acquisition of syntactic knowledge must pass a stiff set of evaluation criteria. Plainly it must actually acquire syntactic knowledge; the model must be powerful and sophisticated enough in either its initial knowledge of syntax, its acquisition procedure, or both, to learn new rules of the sort that people apparently learn. For Lparsifal, this amounts to its initial possession of a full Parsifal interpreter, as well as a detailed knowledge of the proper format for grammar rules context-free base rules, and initial lexical categorization grounded on prior semantic-syntactic correspondences, roughly that Nouns denote Objects in the world, and Verbs actions.[22] On the other hand, this "given" structure cannot be too rich: providing Lparsifal with a full set of instantiated base rules and grammar rules of English would simply beg the question of acquisition, while at the same time leaving unexplained the fact that children from other language environments learn other rules. Any acquisition procedure must therefore tread a thin line between an approach that is constrained enough to account for acquisition (what Pinker dubs "learnability"), yet not so constrained as to rule out observed variation in human syntactic knowledge (Pinker's "equipotentiality" condition).

The dual criteria of "learnability" and "equipotentiality" reflect a model's ability to characterize in an abstract, generative sense the class of possible human grammars. In addition, Pinker has isolated a set of criteria that focus upon a model's plausibility as a reflection of human processing demands. If possible, the output of the acquisition procedure should be a characterization of the process of passing from specific strings of words to their analyzed forms (or vice-versa). In addition, the acquisition procedure should assume an initial state, input data, and available computational resources (time and space) that are least broadly compatible with those that can be reasonably assumed available to (developing) children. The acquisition procedure should not take an infinite amount of time; should use only input data (examples) known to be accessible to children; should approximate the ontogenetic course of acquisition (the order and types of rules acquired and corresponding mistakes

---

22. This correspondence has been widely assumed in the literature. For an explicit formulation, see Limber [1973].

should correspond to observation); and should not demand "superhuman" perceptual or cognitive abilities.

Given these criteria, let us review the success of the Lparsifal model.

1.2.1 Learnability and Equipotentiality

How well does the procedure fare in actually acquiring parsing rules? Starting with *no* grammar rules, the currently implemented procedure acquires from positive example sentences many of the grammar rules in a "core grammar" of English originally hand-written by Marcus. The currently acquired rules are sufficient to parse simple declaratives, much of the English auxiliary system including auxiliary verb inversion, simple passives, simple *wh*-questions (e.g., *Who did John kiss?*), imperatives, and negative adverbial preposing. Carrying acquisition one step further, by starting with a relatively restricted set of context-free base rule schemas -- the "X-bar" ($\bar{X}$) system of Chomsky [1970] and Jackendoff [1977] -- the procedure can also induce the proper ordering for some of the phrase structure rules for English, including some recursive rules. Acquired base rules include those for Noun Phrases, Verb Phrases, Prepositional Phrases, some Complements to verbs, and a substantial part of the English auxiliary verb system. In short, the procedure can acquire a knowledge sufficient to parse an infinite number of sentences on the basis of very simple triggering evidence alone. (For two quick examples of the procedure in action, see Section 1.4.)

This short list of acquired rules is not, of course, even close to a "complete" knowledge of English syntax. Many rules lie beyond the current procedure's reach. Some of these gaps in acquisition are the result of corresponding gaps in computational machinery, while others probably signal a deeper deficit in the grammar-as-parser approach. As an example of an easily-remedied deficit of machinery, consider movement rules. At present, Lparsifal has only a single device to handle all constituent movements. Lacking a distinguished facility to keep track of *wh*-movements, Lparsifal cannot acquire the rules where these movements interact with Noun Phrase movements. (Parsifal employed dual mechanisms to distinguish Noun Phrase and *wh*-movements.) Current experiments with the system include adding the distinguished *wh* machinery back into the domain of acquisition. Similarly, rules to handle conjunction cannot be dealt with until some rules for conjunction are developed by hand for the original Parsifal.

More generally however, the present model cannot capture *all* "knowledge of language" in the sense intended by generative grammarians. For example, since the weakest form of the acquisition procedure does not employ backup, it cannot re-analyze certain sentences as people appear to do, and

so deduce that they are grammatically well-formed.[23]

1.2.2 Psychological Fidelity

The Lparsifal procedure has also been deliberately designed so as to meet (more or less) the remaining criteria of model adequacy. By definition, the characterization of syntactic knowledge attained by the acquisition procedure is a parser, and can be considered a model of the process of mapping sentence strings to their structural descriptions. By fiat, the acquisition procedure uses only positive (grammatical) examples as evidence on which to base the construction of its new rules. This reliance on positive-only examples can be justified on empirical and methodological grounds. Although the final psycholinguistic evidence is not yet in, children do not appear to receive negative evidence as a basis for the induction of *syntactic* rules. That is, they do not receive direct reinforcement for what is not a syntactically well-formed sentence -- so-called negative evidence.[24] If syntactic acquisition can proceed using just positive examples, then it would seem completely unnecessary to move to an enrichment of the input data that is as yet unsupported by psycholinguistic evidence. There is yet another reason for rejecting negative examples as a source of evidence: from formal results first established by Gold [1967], it is known that by pairing positive and negative example strings with the appropriate labels "grammatical" and "ungrammatical" one can learn "almost any" language. Thus, enriching the input to admit negative evidence broadens the class of "possibly learnable languages" too far, admitting systems of rules that cannot possibly be characterized as human.

What about the restrictions on time for acquisition, or on developmental and cognitive capacity?

---

23. For instance, such an acquisition procedure could never determine that the "garden-path" sentence, *The horse raced past the barn fell* is grammatical. Additional machinery must be added in order to accomodate such behavior.

24. One might in fact distinguish at least two types of "negative evidence": (1) explicit negative information: the (perhaps methodical) pairing of positive (syntactically well-formed) and negative (syntactically ill-formed) sentences with the appropriate labels *well-formed* and *ill-formed*; perhaps including correction of ill-formed utterances (alternatively, parses) via (a) explicit negative reinforcement (e.g., *That's wrong*) or (b) tacit negative reinforcement (e.g., responding with the correct pattern, or not responding) and (ii) *indirect negative evidence*, the inference that if a linguistic process P can appear in simple sentences and is not observed to appear, then P does not occur in any sentence, no matter how complex. Chapter 3 shows that every case of indirect negative evidence can be reduced one where only positive evidence is used.

In certain restricted settings adult reinforcement patterns have been observed and might indeed aid acquisition -- they help to overcome the limitations of attention and memory that might be expected of a primitive initial system. But this does not remove the necessity of a highly specified structure within which the reinforcement operates: only by reducing acquisition to the simple setting of a finite number of "parameters" (a finite domain) does simple reinforcement become plausible.
For additional remarks, see also Chapter Two, Section Two; Wexler and Culicover [1980]; for evidence that children do not receive reinforcement for sytnactic well-formedness, see Brown and Hanlon [1970]; or Newport, Gleitman, and Gleitman [1977].

However, children might (and seem to) receive negative evidence for what is a semantically well-formed sentence. This sort of input might prove to be of value in acquisition; for further discussion, see Chapter Two, or Pinker [1980]; Brown and Hanlon [1970].

Here, although sharp statements of convergence time and fidelity to human language ontogenesis are not yet forthcoming, one can show that Lparsifal meets weak upper bounds on these conditions. For instance, any mature parser consists of a fixed Parsifal interpreter and a varying set of grammar and base rules. But the base rules are finite in number, since they can be expressed by a finite set of strings each of finite length, drawn from the fixed and finite theoretical vocabulary of the X-bar theory. Likewise there are a finite number of grammar rules, for the patterns and actions that form all such rules must by convention be expressible as strings finite in length, using a finite vocabulary of symbols. As a result, there are but a finite number of possible Lparsifal parsers (the crossproduct of the number of base and grammar rules). If this set of parsers is descriptively adequate, then this fact by itself ensures that acquisition "in the limit" can be avoided; a simple-minded acquisition procedure that guesses parsers by enumerating all of them (changing a single rule at a time) could arrive at a "correct" parser without taking unbounded amounts of time.[25]

Further, the procedure is deliberately designed to be able to glean more information from simple example sentences (of limited embedding) than more complex examples. This is accomplished in two ways. First, the acquisition procedure cannot be recursively invoked: if, during the process of acquiring a new grammar or base rule the procedure discovers that it must acquire yet another new rule, the attempt to acquire the first new rule (as well as the second) is abandoned. This ban on recursion has the beneficial effect of keeping the procedure's new rules "close" to those it already knows, imposing a certain incremental and conservative character on its developmental history. Second, the grammar rule patterns themselves are restricted under the conventions of Marcus' Parsifal to examine only local context in order to decide what to do next. Together these restrictions mean that if an input datum is too complex for the acquisition program to handle at its current stage of syntactic knowledge, it simply parses what it can, and ignores the rest. The outcome is that the first rules to be acquired handle simple, few-word sentences and expand the basic phrase structure for English; later rules deal with more sophisticated phrase structure, alterations of canonical word order, and embedded sentences. Although this is clearly part of the desired result -- current parsing abilities suggest new rules of parsing, as discussed in Limber [1973] -- it is a picture without much detail. An important area for future research will be to investigate more thoroughly the patterns of over-and under-generalization implicit in the model, comparing them to observed patterns of human language ontogenesis and error production.

---

25. This is, of course, but a casual demonstration that should properly be made formally. However, the "proof" that a finite search space allows an enumerative acquisition approach to converge in finite time is only of technical interest. It seems most unlikely that acquisition actually proceeds by enumerating entire grammars (or parsers): in any event, the time required to enumerate even modest-sized rule systems would be enormous (as discovered by those experimenting with such models: see Horning [1969]; Biermann and Feldman [1972]).

1.3 The acquisition procedure is simple

As mentioned, Lparsifal proceeds attempting to parse a series of positive (syntactically well-formed) example sentences, sentences such as *who did Sue kiss*; or, *I gave the bottle to the baby.* Parsing simply means executing a series of tree-building and token-shifting grammar rule actions. Actions, in turn, are triggered by matches of rule patterns against features of tokens in a small three-cell constituent look-ahead buffer and the local part of the annotated surface structure tree currently under construction.

Grammar rule execution is also controlled by reference to base phrase structure rules. To implement this control, each of the parser's grammar rules are linked to one or more of the components of the phrase structure rules. Then, grammar rules are defined to be eligible for triggering, or *active*, only if they are associated with that part of the phrase structure which is the current locus of the parser's attentions; otherwise, a grammar rule does not even have the opportunity to trigger against the buffer, and is *inactive*. This is best illustrated by an example. Suppose there were but a single phrase structure rule for English that gave the canonical order for a sentence as a Noun Phrase followed by a Verb Phrase, e.g.:

Sentence-->Noun Phrase Verb Phrase (S⇒ NP VP)

Flow of control during a parse would travel left-to-right in accordance with the NP--VP order of this rule, activating and deactivating potentially-matching bundles of grammar rules along the way. To illustrate, we start up the parse by activating any rules associated with the "S" packet. These are grammar rules that are always active, rules whose trigger patterns watch for the characteristic signs of English Sentences -- Noun-Verb clusters such as *Sue kissed...*; *for-to* combinations like *For Sue to kiss...*, and so forth. Now suppose the parser actually detects one of these patterns in the input string (a grammar rule fires to this effect), and enters the S-->NP VP phrase structure rule. First, all grammar rules associated with the "S" remain active, since they must always be ready to trigger (even in the midst of assembling a new Noun Phrase). Second, all grammar rules associated with building the Noun Phrases at the head of a Sentence (in English, the Subject Noun Phrases) would now become active: this would include rules whose job it is to watch out for the tell-tale "leading edges" of Noun Phrases (such *the, any*), and rules specially designed to recognize proper names as Nouns, among others. The parser now has the chance to build a Noun Phrase constituent, eventually advancing in order to construct a Verb Phrase.[26] This step is marked by deactivating the Noun Phrase building

---

26. Thus, once a sequence of constituents is (partially) recognized by triggering on the input stream (by definition, a data-driven or bottom-up process), the parse proceeds depth-first by attempting to recognize completely each of the elements of the sequence in turn.

grammar rules and activating any grammar rules associated with constructing Verb Phrases.[27] Together with (1) the items in the buffer and (2) the leading edge of the parse tree under construction, the currently pointed-at portion of the phrase structure forms a triple that characterizes the current state, or instantaneous description of the parser.

If in the midst of a parse no currently known grammar rules can trigger, acquisition is initiated: Lparsifal attempts to construct a single new executable base phrase structure expansion, lexical insertion grammar rule, or transformational-like grammar rule. New rule assembly is straightforward. Lparsifal first tries to see whether the first item in the buffer can be attached to the parse tree under construction in compliance with the conventions of the X-bar theory. If so, it has constructed a new part of the base component, either a lexical insertion rule or a new phrase structure expansion. If the X-bar restrictions cannot be upheld, Lparsifal attempts to build a new transformational-like grammar rule. To do this, Lparsifal simply selects a new *pattern* and *action*, utilizing the current instantaneous description of the parser at the point of failure as the new *pattern* and one of three primitive (atomic) operations as the new *action*. The local context consists of two parts: (i) a left-context, the top element in the active node stack, plus additional (finite) information about the parse tree constructed so far; and (ii) the right-context, a finite amount of information in the right-hand input buffer. The primitive operations are: *switch* (exchange) the items in the first and second buffer cells; *insert* one of a finite number of lexical items into the first buffer cell; and *drop a trace* (an anaphoric-like element, in the simplest case, an item that must be co-indexed with another Noun Phrase) into the first buffer cell. These actions have turned out to be sufficient and mutually exclusive, so that there is little if any combinatorial problem of choosing among many alternative new grammar rule candidates. Finally, if it arrives at a successful new grammar rule (of either the lexical insertion or transformation-like variety) the system determines whether it must generalize its new rule with an existing grammar rule that performs the same action in the same packet (same point in the base phrase structure) as the new one. If the *attach* action is involved, then the effect of generalization is to form new equivalence classes of lexical tokens. In this case, class splitting is also allowed, if a formerly merged class is shown to require division. Chapter 3 demonstrates that this process of rule generalization and splitting corresponds in simple cases to the induction of a finite state automaton from positive examples, and in complicated cases, to the induction of simple tree automata.[28]

As mentioned, a further constraint is that the acquisition procedure itself cannot be recursively invoked; that is, if in its attempt to build a single new executable grammar rule the procedure finds

---

27. This scheme was first suggested by Marcus [1980, page 60] and implemented by Shipman [1979]. The actual procedure uses the X-bar phrase structure schemas instead of explicitly labelled nodes like "VP" or "S"; see Chapter 2 for details.

28. Lexical classes are therefore defined *dynamically*, by their insertion contexts as determined by grammar rules. There are no categories such as *Determiner*, *Modal*, or *Noun* as such, but rather the classes determined by an invariance relation defined over the behavior of the parser.

that it must acquire still other new rules, the current attempt at acquisition is immediately abandoned.[29]

1.4 Simple scenarios show the procedure in action

To illustrate how the acquisition procedure works, the following two sections present much-simplified scenarios for (1) the acquisition of a context-free base rule to handle certain English Verb Phrases and (2) the acquisition of a grammar rule to deal with inversions in certain English questions. Full details appear in Chapter 2, Sections 5 and 6.

1.4.1 Acquisition of a Verb Phrase expansion rule

In the X-bar theory, all base phrase structure rules for human grammars are assumed to be expansions of just a few templates of a rather specific form, roughly, XP⇒...X..... Here, the "X" stands for an obligatory phrase structure *category* (projected from the features of lexical items such as Nouns or Verbs); in most accounts, the ellipses stand for "maximal projections" of other categories or specified grammatical formatives, such as Articles, Adjectives, or Prepositional Phrases, among others. Actual phrase structure rules are fleshed out by choosing a particular lexical category as the "X" (also called the head of the rule) and settling upon some way to fill out what may precede or follow it. For example, by setting X = N(oun) and allowing some other "XP" to the left of the Noun (call it the category "Determiner" or "Specifier") we would get one possible Noun Phrase rule, NP⇒Determiner N. Or consider the English Verb Phrase rule: here, the Object Noun Phrase is attached to the right of the Head of the rule (the Verb). Other choices give other languages; in German, for example, the object Noun Phrase can be argued to be attached to the left of the verb.[30] Given this template-filling approach, the problem for the learner is essentially reduced to figuring out what items are permitted to go in the slots on either side of the "X". Note that the XP schema tightly constrains the set of possible hierarchical tree structures generated by the base phrase structure rules; for instance, no Noun Phrase rule of the form Noun Phrase⇒ Article Adjective Noun Noun would be admissible as a core part of phrase structure.[31] To see in outline how the X-bar constraints can simplify the phrase structure induction task, suppose that the acquisition procedure has already

---

29. Obviously, a later attempt at the same sentence could lead to different results, if in the meantime the additional missing rule had been acquired by exposure to some other example sentence.

30. Thiersch [1978].

31. Several constraints on the form of phrase structure rules have not been discussed here. For extensive discussion of the X-bar restrictions adopted in this report, see Chapter 2, section 4.

acquired a phrase structure rule for English main sentences,[32]

Sentence⇒ Noun Phrase Verb Phrase

and now requires information to determine the proper expansion of a Verb phrase:

Verb Phrase⇒ ???

The X-bar theory cuts through the maze of possible expansions for the right-hand side of this rule. Assuming that Noun Phrases are the only other known category type, the X-bar theory tells us that there are only a few possible configurations for a Verb Phrase rule:

Verb Phrase⇒ Verb | Verb Phrase⇒ Noun Phrase Verb|

Verb Phrase⇒ Verb Noun Phrase | Verb Phrase⇒ Noun Phrase Verb Noun Phrase

.

.

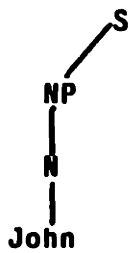
(other combinations of multiple Noun Phrase constituents)

Now suppose that the learner can classify word tokens as Nouns, Verbs, or "other" (perhaps by at first linking items to some semantic grounding as objects and actions).[33] Then, by simply matching an example sentence such as *John kissed Mary* against the array of possible phrase structure expansions, the correct Verb Phrase rule can be quickly deduced; only one context-free tree can be fit successfully against the given input string. Omitting much detail (see Chapter 2), the parse proceeds left-to-right with John first built as a Noun Phrase and recognized as a valid part of the expansion of the already-known rule S⇒NP VP. (Note that *John* is attached as a Noun by a lexical insertion grammar rule, as discussed earlier.) The resulting tree is:

---

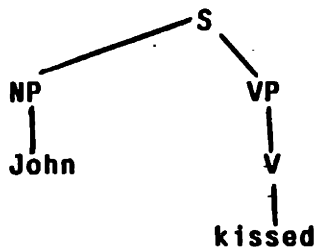
32. No provision has been made so far to account for the symbol "S" appearing in phrase structure rules -- it is not a lexical category. There are two obvious alternatives: (1) there are special phrase structure rules (with constraints of their own) standing outside the X-bar system, to deal with S; (2) S is really some "X" category, most often assumed to be some projection of the category Verb. Both positions have been maintained: (1) Hornstein, [1977]; LaPointe, [1980]; (2)Jackendoff [1977]; Marantz, [1980].
In this report, S will be assumed to be expanded by a rule like NP TNS VP (or NP INFL VP), thus opting for position (1), with INFL as the Head of S. See Chapter 2 for additional disucssion.

33. As Chapter 2 will show, it is quite easy to extend this rudimentary initial categorization into a full set of lexical categories for English, including Adjectives, Adverbs, Articles, Quantifiers, Modal verbs, Prepositions, and Particles.



words to come: kissed Mary.

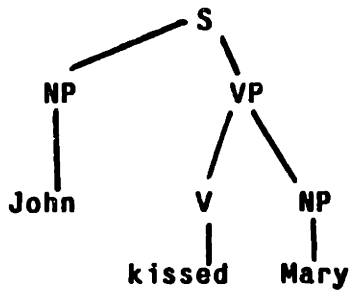
Now, since "kissed" is supposedly recognizable as a Verb, the VP X-bar schema can be entered. The Head of the X-bar schema just entered must, by definition, be a verb; only "kissed" meets this criterion. Consequently, "kissed" can be attached as the "V" portion of the growing tree (again by a lexical insertion rule):



words to come: Mary

(Note that one might have required that the insertion of *kiss* be made contingent upon the presence of a Noun Phrase to the right of *kiss*. This test can be automatically accomodated in the framework of Parsifal via the ability to use the look-ahead buffer. One temporarily suspends processing of *kiss* to analyze the tokens to its right into (possibly) a complete Noun Phrase.)

At a stroke, the options that attach Noun Phrases to the left of the Head in a Verb Phrase are ruled out. The only choice left open is whether the given input example is compatible with Noun Phrase attachment to the right of the Head. The next move of the parser shows that it is: *Mary* is recognizable as a name, hence a Noun and Noun Phrase; it is available for attachment to the Verb Phrase under construction, completing construction of the parse tree:



As can be seen, the VP expansion is as desired. A single simple example has provided positive evidence sufficient to establish a major English phrase structure rule. Although this is but a simple example,[34] it still illustrates how phrase structure rules can be acquired by a process akin to "parameter setting": given a highly constrained initial state, the desired final state can be obtained upon exposure to very simple triggering data. The additional examples in Chapter 2 show how the entire phrase structure system can be reduced to the fixing of just six or so basic parameters within the X-bar framework. From this point of view, there are no non-terminal categories as classically defined, nor any re-write rules that make up a phrase structure "grammar"; rather, these are replaced by the projections of lexical entries. Chapter 2 also probes an exploratory use of the X-bar templates as part of a theory of lexical acquisition. Finally, by establishing an order in which parameters may be fixed, the X-bar theory also delimits the possible "developmental envelopes" for acquisition, providing a theoretical framework for interpreting actual ontogenetic data.

1.4.2 A Subject-Auxiliary Verb Inversion grammar rule

Besides so-called "base" rules, where arguments are supplied to predicates in what amounts to a canonical order, natural grammars also include surface forms where arguments have been displaced from their natural position, as well as variant in the standard surface order of constituents. For example, the sentences, *Who will Sally kiss?* and *Will Sally kiss John* exhibit these two kinds of variations. How can this knowledge be acquired?

---

34. Among other matters, it ignores the so-called "non-configurational" languages, those that do not (apparently) obey the same co-occurrence restrictions as English or German (and which may not even have VP nodes). These might be dealt with via the introduction of a system of phrase structure rules for "S" nodes outside the X-bar theory itself, as suggested by Lapointe [1980].

Suppose that Parsifal has all the base rules and grammar rules to parse *John will kiss Mary.*[35] Now suppose it is given the sentence *Will John kiss Mary?*. No currently known rule can fire, for all the rules in the phrase structure component activated at the beginning of a sentence will have triggering patterns looking for a Noun Phrase followed by a Verb, but the input stream will hold the pattern **[Will: +Verb +Tense] [John: Noun Phrase]**, and so thwart all attempts at triggering a grammar rule.[36] A new rule must be written. The acquisition procedure first tries to *attach* the first item in the buffer, *will*, to the current active node, S(entence) as the Subject Noun Phrase. The attach fails because of category restrictions from the X-bar theory; since *will* is not marked as a Noun, it cannot be attached as the core of a Noun Phrase. But switch succeeds, because when the first and second buffer positions are interchanged, the buffer now looks like **[John][will]**.[37] Since the ability to parse declaratives such as *John will kiss...* was assumed, Noun Phrase-attaching rule will now match. Recording its success, the procedure saves the switch rule along with the current buffer pattern as a trigger for remembering the context of auxiliary inversion. The rest of the sentence can now be parsed as if it were a declarative.[38]

Later example sentences of this type can be shown to quickly generalize the required trigger pattern for inversion to something like, [Auxiliary verb][Noun Phrase]. Here, the label "Auxiliary verb" is actually a gloss for a complex category that is formed dynamically as an equivalence class of items that

---

35. Note that this does not mean that such a system could produce such a sentence. For example, children's use of (unstressed) *do* in declaratives apparently appears *after* its occurrence in yes-no questions (Fletcher [1979 page 272]). On the other hand, *do* sometimes appears as a main verb before its occurrence in the inverted forms (and might therefore be presumed known as marked + Verb). In any case, the early (2:3) appearance of sequences such as, Adult:*Shall I put your watch on?*, Child: *no, I will* (Fletcher, [1979, page 267] from Leopold [1949]) at least suggests that rules exist to handle the declaratively-positioned modal (or auxiliary?) verbs at or about the time the inverted forms are to be parsed.

36. More accurately, *will* will be marked [+ Verb -Noun +Head -Arg] as a member of a more abstract "verb like" category that includes the modals such as *could* or auxiliary verbs such as *have* or *do*, but excludes main verbs. There is, of course, considerable complexity to the English auxiliary verb system that is being glossed over here: auxiliaries and modals can appear only in certain orders, and in certain combinations. See Chapters 2 and 3 for more extensive discussion on the acquisition of these relationships.

37. This procedure clearly presumes that an entire consituent (in this case, a Noun Phrase) has been made available for switching into the first buffer cell. Following Marcus [1980 page 175], the parser does this by temporarily "shifting its attention" to the processing of the Noun Phrase starting with the token *John*. In the case at hand, the Noun Phrase analysis is simple. But since English Noun Phrases may themselves contain Sentential forms (hence other Noun Phrases), this approach leaves open the possibility of an infinite forward chain of attention shifts (hence look-ahead). There are several obvious restrictions that deal with this problem. One (adopted by Marcus) is to note that no "plausible" (or: descriptively sufficient) grammar rule for English ever requires more than five buffer cells total; any shifting beyond this local radius is prohibited (and leads to apparent processing difficulty). This restriction will be called the total buffer cell limitation; it is also adopted in this report. Secondly, one might deliberately disallow recursive invocation of the acquisition procedure; this would push most complex Noun Phrases beyond the reach of the early parser, the recursion limitation. This restriction is also adopted here. The relationship between these two restrictions awaits further investigation. However, both seem independently necessary. Since the buffer cell limit is required by even a "mature" parser, it seems clear that the first restriction is not entirely subsumed by the second; the no recursion condition also appears warranted for other, independent reasons, again to ensure locality of acquisition.

38. The fact that a *switch* was performed is also permanently recorded at the appropriate place in the parse tree, so that a distinction between declarative and inverted sentence forms can be maintained for later "semantic" use. In general, the "active" node, the one currently being built, is annotated with the names of the rules that build it.

do not take NP complements, are partly verbal, and are not Nouns. Importantly, it can also be shown that the grammar rule so acquired also demands that there be no cyclic node immediately above the current active node (in this case, no cyclic node above an "S" node) at the time the inversion is performed. Thus, the Subject-auxiliary inversion rule will not erroneously trigger in an embedded environment -- just as desired. (The ungrammatical sentence, *I wonder did Sue kiss John* should be unparsable given this a set of grammar rules for English.) Note how this example of generalization also conforms to the Subset Principle: inversion is not over-generalized to the case of embedded contexts since no evidence is encountered that would trigger such a move.

Summary of Key Ideas and Accomplishments

o A working computer program has been developed that can acquire substantial syntactic knowledge of English under restrictions faithful to what is known about human acquisition (simple, positive-only example sentences). This knowledge is acquired in the form of an explicit process model of language use (a parser) during the on-going, left-to-right analysis of individual sentences. The syntactic knowledge acquired is of three sorts: order of constituents for context-free base rules, lexical classes, and so-called grammar rules that handle forms such as passive and *wh*-movement.

o The model shows how a transformational theory can be embedded in a theory of language use and language acquisition.

o The acquisition process (rule acquisition) is driven by the parser's attempt to interpret example sentences. It thus provides an explicit "learning theory" for the acquisition of knowledge of syntax.

o Non-instantaneous principles of acquisition have been shown to play a role in formulation of constraints on natural rule systems. A single principle, the Subset Principle, has been implicated in a variety of the constraints exploited by this and other models of acquisition from positive-evidence.

o The procedure required for acquisition is by itself trivial. It is constraints on representations rather than any sophisticated, general learning "heuristics" that play the key role in the success of the acquisition model.

o The interaction between the functional demands of learnability and parsability have been studied in a general automata theoretic setting and using the specific models of acquisition and parsing proposed here. The same locality principles that ensure efficient parsing -- the no-backtracking and bounded look-ahead conditions imposed by Marcus -- also play an important role in acquisition. Bounded look-ahead helps ensure that the "search space" for new rule hypotheses is finite (in fact, small); the no-backtracking condition (Marcus' Determinism Hypothesis) helps pinpoint the exact place in the analysis of a sentence to build a new rule of parsing.

o A simple version of the X-bar theory has been advanced as a theory of the acquisition of base phrase structure rules. The X-bar theory has also proved useful as a model for lexical acquisition and as a source of testable hypotheses about the actual developmental course of human acquisition. The theory is a modular one, in which core cases are determined by fixing a small set of parameters as determined by UG, and marked cases are

determined by brute force via tree automata induction. This combination of acquisition procedures eliminates the heavy data demands of complete reliance on inductive procedures. A system of context-free re-write rules is eliminated as an explicit representation of the knowledge that is acquired when the system "fixes" a grammar. Likewise, traditional lexical category classes (N, V, etc.) are eliminated.

o The Marcus parser has been given an automata-theoretic formalization as a restricted, two-stack deterministic push-down machine. Such a device operates according to an underlying extended LR(k) grammar, as described by Knuth [1965], thus providing a formal characterization of Marcus' Determinism Hypothesis.

o The locality principles proposed for the acquisition model mirror the structural constraints advanced in several current linguistic theories of grammar, as well as the finite error detectability condition of Wexler and Culicover [1980]. The Wexler-Culicover theory is given formal language theoretic treatment, and it is demonstrated that the Finite Error Detectability condition is subsumed by Marcus parsability.

o The constraints of developmental sequencing have been analyzed in the framework of incremental models of acquisition based on positive-only evidence. In certain specific linguistic settings it appears that earlier developmental decisions can constraint later acquisition choices, thus providing an additional source of constraint for the identification of rule systems. This "parametric" view of acquisition may account for apparent developmental stages observed in child language acquisition.

1.5 An outline of things to come

The remainder of this thesis takes up the basic points outlined in this introduction in more detail:

Chapter Two discusses each of the components of the acquisition model: the initial state of knowledge that is assumed; the input data used to acquire new rules; and the acquisition procedure itself. It also motivates the constraints on grammar rule actions described in Chapter One, and then probes the range of the acquisition procedure's capabilities via several examples of the acquisition procedure in action.

Chapter Three presents a formal theory of evaluation measures, using the notion of program size complexity. It applies this theory to the acquisition of base phrase structure rules and phonological rules systems. in a "parameter setting" view of acquisition, then proceeds to several scenarios illustrating the acquisition of base phrase structure rules. As a detailed case study, it examines how the English auxiliary verb system could be examined via a simple finite-state induction procedure, generalizes that procedure by means of tree automata, and then shows how in fact the induction of phrase structure is better modelled as a process of several interacting linguistic constraints, rather than a general inference procedure.

Chapter Four discusses the trade-off between acquisition and recognition complexity in a general formal setting.

Chapter Five presents a formalization of the Marcus parser. It proceeds to formalize the Wexler-Culicover BDE property as a property of grammars, extends the definition of LR(k)-ness to non-context-free languages and examines the intimate relationship between efficient parsability (with respect to Marcus parsers) and Wexler-Culicover learnability.

Appendix I - A Listing of Rules Currently Acquired

Grammar Rules in Parsifal and Grammar Rules Acquired

Total # of Rules in Parsifal[39] = 107, of which 83 are purely "syntactic" rules.
Total # of Rules that can be currently acquired = 58, about half of all the rules, or three-quarters of the syntactic rules.

Rules by Type

Transformational Rules: 10 Acquired: 8

1. Passive (*A meeting was scheduled for Tuesday.*)
2. Auxiliary verb inversion (*Was a meeting scheduled for Tuesday?*)
3. *there*-insertion (*There was a meeting scheduled for Tuesday.*)
4. (Simple) *Wh*-gapping (*Who scheduled a meeting for Tuesday?*)
5. Imperative (*Schedule a meeting for Tuesday.*) (Note: hand-simulated only)
6-8. *Pro* insertion in embedded clauses (3 rules)
(*I promised to schedule a meeting.*)
9-10. *To, be* insertion in embedded clauses (2 rules)
(*I want a meeting scheduled for Tuesday.*)

Acquired:
All of the above except for 2 of the Pro rules.

Phrase Structure Rules: 34 Acquired: 32

Main Sentence Phrases - 4 rules
Auxiliary Verb Phrases - 10 rules
Verbs - 4 rules
Prepositional Phrases - 3 rules
Verb Complement Clauses:
That-complements (*I knew that a meeting would be scheduled.*) - 3 rules
To-infinitives (*I want to schedule a meeting.*) - 2 rules
Object Phrases - 3 rules

---

39. All rules dealing with number expressions, clock times, and the like, hand-tailored for the personal assistant project, are not included in this count.

Embedded Sentential Clauses - 2 rules
Labelling rules - 3 rules

Acquired: All of the above rules, except 1 of the *that*-complement rules, and 1 of the Verb rules.

Noun Phrase Rules: 24 Acquired: 11
Start Noun phrase - 2
Determiner - 1
Quantifier - 4
Adjective - 1
Noun - 1
Pronoun - 1
Proper name - 3
Other modifier - 3
Incomplete NP - 8

Two of the quantifier rules, the three "other modifier" rules, and the incomplete NP rules cannot be acquired.

Wh-phrase Rules: 7 Acquired: 4

Create *wh* phrases - 2
Reduced relative phrases - 1
Wh-PP phrases - 2
Finish *wh* phrases - 2

The reduced relative and *wh*-PP rules cannot currently be acquired.

Lexical Diagnostic Rules: 8 Acquired: 3
1. *Have* diagnostic (*Have the boys take/have the boys taken*)
2. *A* diagnostic (*A hundred pound bags/a hundred pound bag*)
3. *Which* diagnostic
4-6. *That* diagnostic (*I know that boy with red hair/I know that John has red hair.*) - 3
7. *To* diagnostic
8. *What* diagnostic

The *have*, *a*, and one *that* diagnostic can be acquired.

The following rules require "goodness of fit" evaluations provided by the case frame interpreter, and, as such, cannot be acquired by the current system.

Wh "semantic" rules - 6
Other "case frame attachment rules" - 9
Prepositional Phrase attachments - 8
Other case frame - 1

2. The Components of the Acquisition Model

As outlined in Chapter One, the acquisition model of this thesis can be divided into two components:

> (1) The specification of an initial state of knowledge -- what is assumed known in advance of any input from the external world;
>
> (2) The specification of an acquisition procedure that drives the system from its initial to final state. This includes both the specification of a set of input data from which to infer new knowledge and the acquisition procedure itself.

In the first three sections of this chapter we shall outline each one of these components in turn. Section 1 discusses what is assumed about the initial state of the "bare" interpreter; Section 2 reviews assumptions about the input data available to the acquisition procedure, and Section 3 describes the acquisition procedure itself. Section 4 continues with a discussion of the way in which phrase structure rules were implemented in the original Parsifal model, and how these rules may be eliminated in favor of the so-called "X-bar" conventions for phrase structure; the section concludes with a discussion of several acquisition scenarios for base phrase structure rules. The next section of the chapter, Section 5, moves on to examine in detail the induction of the Auxiliary system of English. Finally, Section 6 concludes the Chapter with a discussion of the acquisition of non-base rules, including detailed acquisition scenarios for rules such as Subject-Auxiliary inversion, passive, *wh*-movement, and the like. A complete table of rules currently acquired by the acquisition system is also included, in an Appendix.

To begin, we will discuss each of the components of the acquisition model, starting with a specification of the assumed initial state.

2.1 Component 1: The Initial State of Knowledge

The final state of the parser's knowledge about syntax is simply its ability to analyze sentences, an ability that divides into the interpreter-plus-rules model discussed in Chapter One. We fix an initial state of knowledge by specifying a bare interpreter and an initial set of rules. In the case of Lparsifal, this amounts to taking the bare-bones Parsifal interpreter as a given, along with a knowledge of the proper format for base phrase structure rules, lexical insertion grammar rules, and transformational-like grammar rules. No specific instantiations of rules are assumed known at all. The rest of this section will be devoted to explaining just what this bare-bones interpreter looks like, as well as the format for rules.

Recall that Parsifal acts as an interpreter for grammar rules of a particularly simple pattern-action form. Parsifal is a simple finite-state control device with two stacks that observes features of items in

the input stream and also features of its own making (the partially-built parse tree) and then allows certain actions to be executed. (For a formalization of Parsifal as a two-stack parser, see Chapter 5.) The actions are the basic operations that build the parse tree itself. To illustrate, a (simplified) grammar rule to parse the first portion of the sentence, *Will Sally kiss John?* (Subject-auxiliary inversion) might have the pattern and action as indicated in Figure 2.1 (a) below.

IF	[first item in input is an auxiliary verb (such as *will*) second item in input is a Noun Phrase, (such as *Sally*)]
THEN	[ATTACH the first item as the auxiliary verb of the sentence ATTACH the second item as the Subject of the sentence; LABEL the sentence a yes-no question]

Figure 2.1 (a) - Simplified rule to handle auxiliary verb inversions.

IF	[same pattern as above]
THEN	[same action as above]
ONLY WHEN PARSING	[Subject Noun Phrase]

Figure 2.1 (b) - Same rule with base phrase structure control.

Parsifal was designed primarily to handle syntactic phenomena, producing as output a modified form of the *annotated surface structures* of current transformational linguistics (Chomsky [1975]; Fiengo, [1974]; [1977]). Considered in the abstract, Parsifal is simply a function that takes *strings of words* to *labelled bracketings* (equivalent to parse trees). *Semantic* processing is not strictly within Parsifal's realm.[1]

---

1. However, because the parse tree can be used later on, or, as in Marcus' scheme, for concurrent semantic processing, the syntactic representation can be considered to interact with some hypothesized semantic component. Since in this view the syntactic and semantic components must be "impedance matched" to one another (using now Marcus' terminology), one might expect that the acquisition of syntactic knowledge indeed constrains and is constrained by a hypothetical semantic component. However, for the purposes of this report we will focus almost entirely on the acquisition of syntactic knowledge, invoking constraints from a presumed semantic component when necessary. For instance, the ability to segment the sound stream into individual tokens is almost certainly an important prerequisite for the ability to parse. This issue is further discussed in Section 2 immediately below.

The grammar rules are further controlled by the operation of specific context-free base rules that determine whether a particular grammar rule is permitted to trigger at all. In the case of the sample rule above, we would obviously like the inversion to apply only when parsing the initial portion of a sentence (i.e., in English, the Subject-auxiliary verb portion), and then automatically forbidden later on. Imposing this control would be simple if there were a canonical way to block a sentence into modular "chunks" -- like Subject--Verb--Object -- because then we could use the ordering implicit in the canonical arrangement of the chunks as a way to determine when to "switch off" the possible triggering of grammar rules. For example, the Subject-auxiliary inversion rule could be rendered permanently inactive as soon as we know that we have finished with the Subject-Verb modules. It is the context-free base rules (the so-called base *phrase structure* rules of a language) that furnish just this desired canonical ordering. For instance, one such base rule appropriate for English is that a Sentence consists of an initial Noun Phrase (the Subject Noun Phrase) followed by a Verb Phrase. By attaching a condition to the Subject-auxiliary inversion rule such that it can apply only when the system is attempting to parse an initial Noun Phrase, the additional control is easily achieved; see Figure 2.1 (b).

A parse proceeds simply by making a single left-to-right pass through a given input sentence, with the interpreter executing any grammar rules that happen to match on the environment of the parse (features of the input tokens and portions of the already-built parse tree), ending with an annotated tree as output. The restriction to a single sweep through the sentence was specifically designed by Marcus to reflect the exigencies of human sentence processing.

The syntactic knowledge of Parsifal thus divides into two parts, a basic *interpreter* and the simple *programs* -- grammar rules operating partly under the command of the base rules -- that the interpreter executes. Figure 2.2 immediately below illustrates the division. Given this modularity, the natural way to model the acquisition of syntactic rules is to take the basic operation of the interpreter as fixed, corresponding to an initial set of abilities. Two sorts of rules are therefore acquired: the grammar rules and the base (phrase structure) rules for the language at hand. In short, our goal for English will be an acquisition procedure that can construct grammar rules similar to the Subject-auxiliary inversion rule and base rules similar to the Sentence-->Noun Phrase Verb Phrase rule. For grammar rules, this will mean a procedure that takes as input sample sentences and then assembles a new valid rule out of the proper patterns, actions, and only when conditions. For base rules, this will mean a way to determine the canonical constituent arrangement for a particular language, e.g., that English sentences are regularly of the form "Noun Phrase-Verb Phrase" rather than "Verb Phrase-Noun Phrase".

The knowledge initially provided as a foundation for the acquisition of these two sorts of rules includes the basic machinery of Parsifal as well as the vocabulary of pattern descriptions and actions and a crude ability to isolate and label words in the input stream. The following knowledge is assumed:

o **The major data and control structures** of the parser; the utility programs that maintain the data structures and perform routine matching tasks.

o A **dictionary** that, initially at least, can classify input tokens as *noun*, *verb*, or *other* (unknown) (corresponding to the feature system, ±N, ±V). These categories correspond to lexical items that, initially at least, can be determined to be Objects or Actions, respectively. Later on in this Chapter we show how this initial correspondence is relaxed and how additional categorizations can be made.

o A finite stock of **syntactic feature primitives**, tags attached to items in the input stream that serve as predicate tests for Parsifal's grammar rule triggers. For example, the token *Sally* might come attached with the features, *Name, Singular, Animate, Feminine...* (The proper formulation of such a list of primitives is of course open to empirical review; see Keil [1979] for a discussion of the development of children's ontological categories and Chapter 3 for a formal model of how categorization systems might develop.). Besides the features N (Substantive) and V (predicative), the features H (for Head of a phrase) Arg (for an NP Argument), and T (for a Tense Operator) are currently used.

o An *unordered* **template phrase structure rule**, adapted from Chomsky [1970] and Jackendoff's [1977] **X-bar** theory of phrase structure rules. In brief, each phrase is assumed to be of the form, {Head, Specifier, Complement}, with these three components unordered.

o A **rudimentary well-formedness constraint** for the thematic structure of sentences and verbs (and possibly items of other syntactic categories). Sentences must meet an operator-operand constraint, where operator = an Inflectional element, and operand = a complete predicate (a VP), and an NP. Sentences are assumed to be of the form {INFL, NP, VP} (with these constituents initially unordered). In addition, the proper assignment of arguments to verbs is assumed recoverable in simple cases; e.g., *kiss* is presumed to require a Goal or Patient argument and the initial correspondence between this argument and the Direct Object structural positions is assumed to be derivable from "context." (This assumption is comparable to that made in the Wexler-Culicover model of the acquisition of transformational grammars [1980].)

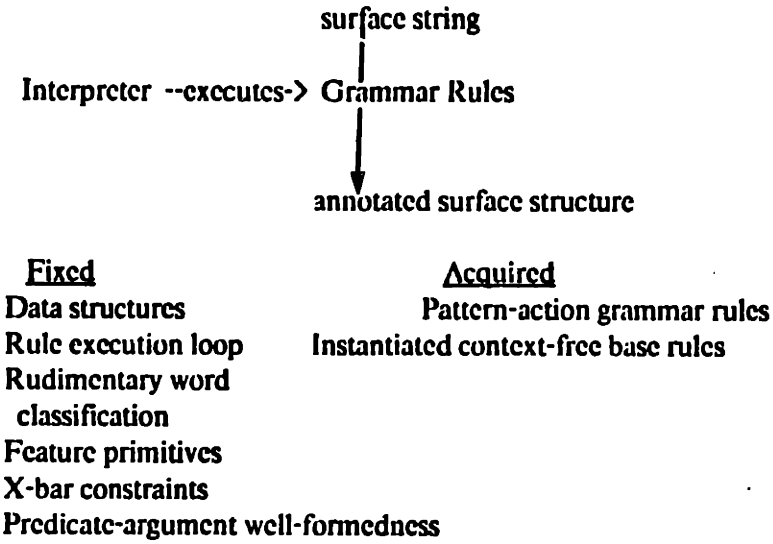


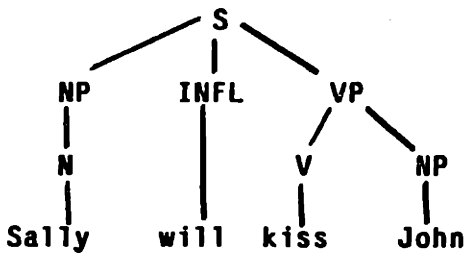
Figure 2.2 - Parsifal's interpreter is fixed; grammar rules and expansions of base rules acquired.

Because the Parsifal data structures form the core set of the "givens" for the acquisition program, it is important to understand in somewhat more detail their role in parsing a sentence. Parsifal is built around two major data structures motivated by the theoretical goal of determinism (tree-building actions cannot be undone) and the more practical goal of building a parsed representation of the input sentence. The output of the parse of *Sally will kiss John* might look like that in Figure 2.3 (a) below. Part (b) of Figure 2.3 shows a snapshot of this same tree while under construction by Parsifal. The emerging tree structure is stored as a stack of constituent nodes, the active node stack. The active node stack is designed to hold the phrases of the sentence in proper hierarchical order for their assembly into a complete parse tree; it is dubbed "active" because the phrases held there are either not yet completely built -- Parsifal does not know all the subtrees (daughters) of that phrase -- or else because that phrase's attachment to some higher (still active) phrase has not yet been determined. In the example at hand, the S(entence) node (labelled "S20") is on the top of the stack and is active because in the snapshot as given the entire sentence has not yet been analyzed -- more daughter phrases, in particular a Verb Phrase, are still to be attached to the Sentence node.[2] Figure 2.3 (b) also shows that two nodes -- a Noun Phrase (NP) node and an Auxiliary Verb (AUX or INFLection) node -- have already been attached to the main Sentence node. The attachment of these two phrases to the Sentence node signifies that by the time this snapshot was taken Parsifal had already determined the

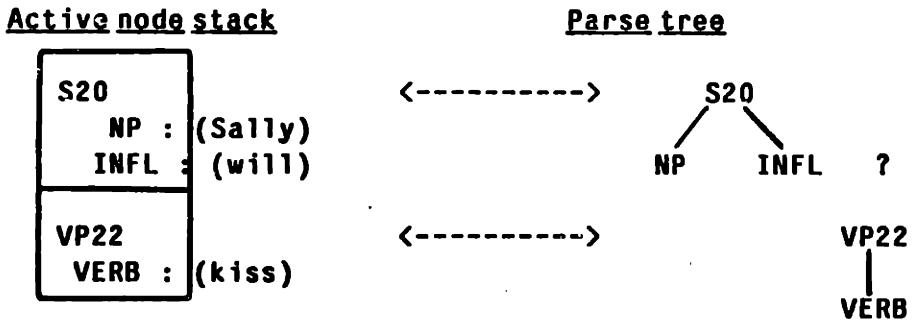
2. Marcus inverted the usual convention that the *top* element of a stack refers to the first accessible item; in Parsifal, it is the *bottom* item on the stack that is the locus for a push or a pop. This was done so as to comport with the graphic convention that a parse tree is built *top-down*, with the accessible frontier of the tree at the bottom.

grammatical role of these two phrases in the sentence. Note that the Noun Phrase and Inflection nodes are not therefore part of the active node stack, but are simply part of the graphic representation of the emerging annotated surface structure tree.

The second (and bottom-most) node of the active node stack in this snapshot is the Verb Phrase node, VP22. Just like the Sentence Phrase, it too is active because it has not yet been completely built -- although the Verb *kiss* has already been attached to the Verb Phrase node, the Noun Phrase object *John* has not been. It is because there are *two* as yet unfinished constituents that we need a *stack* of active nodes. Further, typically only the bottom-most node of the tree (="top of the stack" in standard terminology) can be subject to grammar rule actions. In this respect, the active node stack acts like a pushdown store, maintaining stack discipline with respect to effects of grammar rules upon the active nodes. In Figure 2.3 (b), VP22 at the bottom of the stack has become for the moment the focus of the parser's efforts. That is, the Verb Phrase node has been singled out as the current active node (denoted C), and grammar rules that execute will attempt to build structure under this node. Once the Verb Phrase node is completely built (the object Noun Phrase attached), the parser will attach it to its place in the S node above, removing it from the stack of active nodes, and assigning the S node the status of *current* active node.



(a) Surface structure tree



(b) Emerging node stack representation of tree.

Figure 2.3 - Surface structure tree and snapshot of Parsifal stack representation.

The second major data structure of Parsifal, reflecting the concerns of Marcus' Determinism Hypothesis, is a three-cell constituent buffer. It is the buffer that holds either incoming words from the sentence string, or phrases whose grammatical function has not yet been completely determined. As Figure 2.4 illustrates, each cell in the buffer can hold a single word or several, if these words all lie under a single node (such as a Noun Phrase node).

```
The girls will kiss John.

-----------------------------------
| NP27       | WORD36 | WORD37   |
| THE GIRLS  |  WILL  |  KISS    |
-----------------------------------
```

Figure 2.4 - Parsifal's buffer holds words or constituents.

Parsifal delays deciding about what syntactic structure should be built -- that is, what nodes or tokens to attach to what other nodes in the active node stack -- until it has had the opportunity to use (if necessary) the local context information in the buffer. The Determinism Hypothesis claims that by postponing structure-building decisions in this fashion, no choices will ever have to be undone. The necessity for such a look-ahead facility in a parser that operates left-to-right can be seen from a cursory examination of pairs of sentences like those below, from Marcus [1980 page 15]:

(a) Have the boys who missed the exam take the makeup today.

(b) $[_S[_{VP}$ Have $[_S$ the boys who ... take the makeup today]]]

(a) Have the boys who missed the exam taken the makeup today?

(b) $[_S\,[_{AUX}$ Have$][_{NP}$ the boys who ...$][_{VP}$ taken the exam today]]

To quote Marcus,

> It is impossible to distinguish between this pair of sentences before examining the morphology of the verb following the [noun phrase] "the boys". These sentences can be distinguished, however, if the parser has a large enough "window" on the clause to see this verb; if the verb ends in "en" (in the simple case presented here), then the clause is a yes/no question, otherwise it is an imperative. Thus, if a parser is to be deterministic, it must have some constrained facility for look-ahead.
> [1980 page 17]

The typical effect of grammar rules is to remove items from the first cell of the buffer and attach them to the current active node -- the lowest item on the active node stack. In addition, if the grammatical role of a constituent is undetermined, rules can insert such items back into the buffer. Recall as well that grammar rules fire if and only if their associated patterns match the current instantaneous description of the parser (some combination of features predicated of the items in the buffer and the current active node, plus an active packet). Features are typically grammatical descriptive elements recovered from the lexical retrieval of items in the sentence string or added by the parser's own actions; they include items like number (plural/singular); tense (past/present/future); and verb

subcategorization (transitive/intransitive).[3] Figure 2.5 below displays the pattern and action for the Subject-auxiliary verb inversion rule that was informally presented In Chapter One. The version in the top half of the figure is written in English, while that in the bottom half gives the form quite close to that actually processed by Parsifal:[4]

```
        Rule Pattern
      item examined                  feature to match against
      current active node               Major sentence
      1st buffer cell                    Auxiliary verb
      2nd buffer cell                    Noun Phrase
      3rd buffer cell                    none

      Rule action
   Switch first and second buffer items.
```

(a) Pattern and action for a Subject-auxiliary verb inversion rule.

```
Pattern: Current active node (C)          buffer
                                         1st   2nd  3rd
    [**c; * is S MAJOR]                 [=AUX][=NP][    ]-->

Action: switch
```

(b) Abbreviated form of the same rule.

Figure 2.5 - A typical grammar rule pattern and its abbreviated form.

3. We have not yet said whether these features are themselves primitives or whether they are acquired. Whether a given verb subcategorizes an argument or not is determined by exposure to the right examples, given additional assumptions about X-bar theory and that thematic assignments are recoverable. So for instance, the sentence *Sally kissed Bill* triggers the creation of a subcategorization frame for *kiss* of the form [__NP]. See later in this chapter for additional examples; the subcategorization frame is actually checked via unification with the features of the Head of a Phrase. For example, if the verb in question is *want*, then the VP dominating *want* will have the feature [__Arg], meaning that an Argument (NP or S-bar) is required. Note that this demand does not specify the *string ordering* of the NP or S-bar, whether to the left or right of the Verb -- this is set via a positive example.

In brief then, the transitive/intransitive distinction is acquired on a verb-by-verb basis. What about features such as plural, or tense? Something like a tense operator is assumed given, in the form of the basic NP, Inflection, VP constraint on sentence well-formedness. However, the actual class of tense operators is induced. See later in this chapter and Chapter 3 for details. Whether a distinction based on *number* is primitive or not will not be addressed in this study.

4. The original Parsifal did not have a *switch* operation; its effect was simulated by an *attach second item* grammar rule.

In simple English, the rule in Figure 2.5 says that *if* the first item in the buffer has the feature *auxiliary*, the second is a Noun Phrase, the third item is anything, and the current active node is a *S*(entence), *then* interchange the first and second items in the buffer.

Finally, functionally related grammar rules are grouped into *packets*. It is the packet system that implements what was informally referred to previously as the "only when" conditions for grammar rule execution: the grouping of grammar rules into packets provides a way of controlling whether entire sets of grammar rules should be made available for matching against the buffer items. For example, all the grammar rules that analyze Subject Noun Phrases can be clumped together, and unless this Noun Phrase packet is *activated*, the parser will not even attempt to match the grammar rules in this packet against the buffer. When the parse of a constituent is complete, that packet is *deactivated*. The packeting system is discussed in more detail later on in this chapter. There it is shown how the activation and deactivation process can be formally associated with base phrase structure rules, thereby reducing the acquisition of this information to the acquisition of base rules.[5] It is then demonstrated how the packet system can be completely eliminated in favor of a representational structure based on X-bar theory alone. No information about packets need be learned. Instead, only easily acquired information about the *order* of basic constituents must be acquired.

It should be apparent from the preceding discussion that the behavior of the Parsifal interpreter during any given parse can be completely characterized by a sequence of snapshots that depict the contents of the active node stack and buffer, and the currently active packet. Following the terminology of automata theory, such a snaphot will be referred to as an instantaneous description (or *ID*) of Parsifal. Thus a (successful) parse will consist of an ordered sequence of valid ID's, commencing with an empty stack, buffer, and active packet and ending (if the sentence is grammatical) with a complete parse tree as the only entry in the node stack and an empty buffer and active packet.[6]

To summarize, there are two major data structures crucial to the operation of the Parsifal interpreter. The active node stack provides the usual computational power required for handling the recursive syntactic structures typical of natural languages. The look-ahead buffer serves as a "window" to hold constituents or individual words under current analysis, reducing the parser's guesses about what the

---

5. This approach was first implemented by Shipman [1979].

6. A formal presentation is given in Chapter 5. Briefly, one can define a restricted 2-stack parser as a 5-tuple (Q, L, R, T, P), of states, left-stack symbols, right-stack symbols, terminal symbols, and productions. The right-stack is the analogue of Parsifal's input buffer, the left-stack, Parsifal's active node stack. The productions are limited to moving elements from the top of the right stack (=the left most element of the input buffer) to the top of the left-stack (=the current active node), or vice-versa; using the look-ahead and changing state, or reading new tokens. The system is less general than an unrestricted 2-stack deterministic push-down automaton because (1) it is designed to halt on all inputs; and (2) it possesses certain local error-detecting capabilities (just like the LR(k) parsers discussed by Knuth [1965]).

next parsing decision should be. If Marcus' thesis is correct, a small look-ahead can actually reduce the amount of guessing and back-up to zero, so that a single left-to-right sweep through a sentence will suffice for its correct analysis. Further, the interpreter deals with two different sorts of rules that direct the parse. Grammar rules are unitary pattern-action productions that perform the real work of the parse, actually building tree structure, labelling nodes, and the like. Base rules, corresponding to packets in the original Parsifal design, dictate when certain bundles of grammar rules are permitted to apply. Both these types of rules are to be acquired by the Lparsifal acquisition procedure.

2.2 Component 2: The Input Data

2.2.1 Pre-segmented Grammatical Sentences Provide the Input Data

It has been assumed that the acquisition procedure uses as evidence for its hypotheses about syntactic knowledge only *grammatical* example sentences, so-called *positive* data. Ruled out is the presentation of *ungrammatical* sentences, followed by an indication that the example was ungrammatical, or explicit correction of the learner's syntactic mistakes. Such pairing of ungrammatical sentences followed by an indication of ungrammaticality is usually termed *negative data*. The assumption that only positive data is exploited for the acquisition of syntactic knowledge may well be too strong. On the other hand, most psycholinguistic experiments indicate that it is not (Brown and Hanlon [1970]; Brown, [1973 p. 387]; Newport, Gleitman, and Gleitman, [1977]; summary in Wexler and Culicover, [1980]). Given our still uncertain knowledge about the linguistic evidence input to the child, the assumption that only positive data is employed for the acquisition of syntactic knowledge is the strongest and safest claim we can make. If acquisition can proceed using only positive data, then it would seem completely unnecessary to move to an enrichment of the input data that is as yet unsupported by psycholinguistic evidence.

In somewhat more detail, we suppose that the procedure receives as "input data" a sequence of grammatical surface strings derived from a "mature" syntactic component. For instance, the stream consisting of the three tokens ordered as "Sally"--"kissed"--"John" is one such valid surface string (for English). By "grammatical" one simply means that such strings conform to the principles that constitute a "mature" (i.e., adult) linguistic knowledge. Note also that this approach assumes that the input stream has already been partitioned into individual words (such as *Sally*) and even grammatical morphemes (endings such as the "ed" in *kissed*). Further, individual words are presumed to come uniquely labelled as Noun, Verb, or Other (unknown), and are tagged with a set of syntactic features potentially of value for Parsifal's parsing rules. That is, the input stream *Sally kissed John* is not literally input to the acquisition procedure, but rather is first transformed so as to mark off major lexical categories and tag important syntactic features, roughly as follows:

Sally kissed John

$[_N$Name, singular, ...$][_V$ +ed Past...$][_N$ Name, singular, ...$]$

Figure 2.6 - Input stream and form actually input for acquisition.

Admitting a structured input of this form clearly presupposes that some grasp of individual words (their segmentation and feature labelling) precedes the development of parsing abilities. This is an

idealization demanding some justification. Is it a valid methodological simplification? Does not language acquisition also involve the acquisition of these other capabilities?

2.2.2 Segmentation and Words

The simplifying assumption that a basic knowledge of words precedes more sophisticated abilities at parsing can be supported on several grounds. The first is methodological: it allows work to proceed, permitting concentrated focus on just a single aspect of language acquisition. But this assumption also appears to be logically justified. An ability to *parse* presumes some ability at assigning different words to different parts of a tree (or labelled bracketing). It is hard to see how the notion of a "syntactic rule for parsing" can even be made coherent without some way to pick out individual words from the continuous speech stream. If so, then since this research is about the acquisition of syntactic rules it is a reasonable methodological simplification to take some ability at segmentation as given, since it is logically prior to the syntactic rules to be acquired.[7]

It at least makes sense then that a rough grasp of the categorization features of *some* individual words should precede the development of parsing abilities that make use of such elements. In fact, this appears to be the case for children: production of single words precedes connected speech (although the complicating effects of developing processing abilities suggest the usual likelihood of confounding effects) (Brown, [1973 page 390]). Likewise it is plausible that the initial categorization of lexical items as Nouns or Verbs can be accounted for by the obvious (and often suggested) proposal that nouns can be identified with objects and verbs with actions.[8]

---

7. This is much like the usual assumption about *levels of representation* that has been made since at least the early work of Chomsky [1955]: syntactic rules require for their proper operation an "input language" -- a specific representational format; that representation is logically prior to those rules. For example, if the English plural morpheme "s" is not isolable as a distinct entity by some sort of mechanism, then it would seem highly unlikely, indeed, impossible, that a syntactic rule could make reference to such a representational unit.

8. It should be emphasized that the recent Government-Binding Theory of Chomsky [1981] is actually designed to maintain this close connection between syntactic and semantic representations. Note that in the early child language the correlation between a supposed canonical predicate-argument form and surface order of words is nearly exact: as has been often observed, children produce utterances as if they followed a fixed Agent-Action-Object order (Brown [1973]; Limber [1973]). Adherence to a strict order strategy sets the stage for the detection of deviations from this rigid word order (e.g., inversions in yes-no questions), because everything can be held fixed save for a single deviation. Government-binding theory is helpful in this regard because it assumes that semantic structure, in particular the predicate argument structure of verbs, is "projected" through to the level of annotated surface structure. The idea is, roughly, that sentences "wear their meaning on their meanings on their sleeves" -- if we are willing to given sentences a more abstract representation than their simple labelled bracketing. So for example, the abstract surface form of the passive "Bill was kissed" is designed to parallel the usual declarative form, "Sally kissed Bill." so that thematic relationships can be read directly from syntactic form. In effect, the Government-Binding theory is claiming that simple N-V-N strategies never have to be abandoned, because syntactic form never deviates from semantic form, contrary to surface impressions. This confluence of representational levels has a prima facie appeal from the standpoint of acquisition, of course.

There is a corresponding account of developing word segmentation ability. Once again, we appeal to a principle of inference from clear cases, whereby only unambiguous examples of word segmentation are exploited for the acquisition of new syntactic knowledge. As has been frequently observed, speech to very young children is punctuated by features that would serve to ease the extraordinary burden of separating out individual words in the sound stream: use of single words, special intonation and stress, slow delivery (Brown [1973]; Newport, Gleitman and Gleitman [1978]). Such characteristics would tend to ensure that the developing child gets the sort of input that we have assumed for Lparsifal: a segmented input stream. Once certain clear cases of words have been recognized, then their psychophysical cues might be relaxed. As designed, Lparsifal actually has this property: since its acquisition is incremental, sentences that are "too complex" or that require the simultaneous inference of segmentation boundaries *and* grammar rules are simply left uninterpreted until the proper time comes for their acquisition.[9]

In any case, the assumption of prior development of segmentation and word classification leaves the door open to a more sophisticated theory that specifies a detailed interaction between developing segmentation, phonological, and syntactic abilities. As discussed in the previous footnote, constraints on possible phonological representations could supplement those provided by syntactic representations, providing an additional "forcing function" that aids the learner.

2.2.3 Features and Adult Correction

One assumption about the form of the input data remains: the labelling of words with features drawn from a finite, semantically-grounded stock of primitives such as *Singular* and *Animate* -- those features potentially available to Lparsifal as its basis for constructing triggering patterns for grammar rules. The program "knows" what universal set of features is possibly relevant for its grammar rules simply because it has been told what the members of that set are. How can this be justified?

The puzzle of which features to consider as relevant for the induction of knowledge from an infinitely

---

9. See Brown, 1973, page 390 for discussion. Of course, for this account to be made precise one must supply a more fully-developed theory of adult processing of speech. This is a research topic in its own right. It could be that recent developments in the metrical theory of phonology (the theory of syllabification and weak-strong stress contours) (Liberman and Prince [1977]) could play some role in advancing our understanding of early development of speech segmentation. It is well known, for example, that infants mimic the stress contours of full sentences before producing them complete with words. (A fact that might be due however entirely to unfolding processing capabilities.) The metrical restrictions might presumably aid learning because they provide a narrower characterization of the valid syllabification patterns of human languages, and possibly the boundaries of words and phrases themselves. Acquisition of the syllabification patterns for a particular language could even parallel the template-instantation used by Lparsifal to flesh out context-free base rules. The same sort of "parameter setting" outlook may also provide an account for the unfolding of phonological feature categories; see Kean [1974] and Chapter 3. A complete theory might have to combine a structured, parameter-setting model with a probabilistic and distributional approach to segmentation, as advanced by Olivier [1968].

variable environment is of course just a restatement of the central problem of knowledge acquisition itself. As Pinker has noted (citing Slobin):

> Natural languages dictate that certain semantic features of the sentence referent (e.g., number, person, gender, definiteness, animacy, nearness to the speaker, completedness, and so on) must be signalled in prefixes, suffixes, alternate vowel forms, and other means. However, these features are by no means all that a child *could* encode about an event: the color, absolute position, and texture of an object, the time of day, the temperature, and so on, though certainly perceptible to the child, are ignored by the morphology of languages, and hence should not be encoded as part of the semantic structure that the child must learn to map onto the string. ... Thus there has to be some mechanism in the child's rule-hypothesization faculty whereby his possible conceptualizations of an event are narrowed down to only those semantic features that languages signal, and ultimately, down to only those semantic features that his target language signals.
> [Pinker, 1979, fn. 4, pages 272-273; from Slobin, 1978, personal communication]

One (obvious) reply to this conundrum is that the proper set of semantic features for syntactic rules could be universal, that is, innate (though perhaps in some complicated sense). That would completely solve the problem of narrowing down conceptualizations "to only those semantic features that languages signal", and would justify allowing Lparsifal access to only that set of features for its grammar rules. Work by Keil [1979] indicates that the categorizations children can make unfold according to a constrained process of aborization. Roughly speaking, new and natural ontological categories are split off from old ones; categories than cannot be expressed as the hierarchical refinement of other categories are unnatural. Animacy, possession, multiplicity, and the like seem to be categorization concepts that are acquired relatively early, though of course, there is a wealth of detail to the actual course of acquisition of such concepts, and much that is unknown. It seems reasonable to take these features as given. One must still recognize that this may be too strong an assumption to make. Since a complete answer to this problem would seem to rely on a better understanding of the developing child's (or adult's) conceptual armamentarium -- and thus a theory currently out of reach -- one needs an interim approach to the "encoding problem" to get around this impasse and at the same time settle the remaining problem of narrowing down the set of universal features to those exploited by the language at hand.

A simple-minded proposal is to adopt a corollary of the Subset Principle outlined in Chapter One: guess the narrowest possible language consistent with positive evidence seen so far. In terms of features that trigger when grammar rules can operate, what does this mean? Since additional features can only limit when a grammar rule can be executed, application of the Subset Principle demands that one err on the side of providing *too many* features for the grammar rule pattern. For suppose otherwise, that is, suppose too few features are supplied for the triggering pattern of a grammar rule. Then that rule will be too general; it will execute in too many environments, and hence lead to a larger possible class of surface strings that are acceptable. In short, it will generate too large a language. But

then, given positive-only evidence, the system can never know that it has acquired too large a language.

An over-supply of features is not a severe computational burden, and can quickly be remedied by rule generalization. The details of generalization are covered later in this chapter, but the basic idea is simple. Whenever a new grammar rule is constructed that specifies the same action as an already-known rule(s), and at exactly the same point in the left-to-right parse, *merge* the rules by intersecting the features that the two rules have encoded in their patterns. This has the beneficial effect of quickly pruning away irrelevant features. For instance, suppose that the Subject-auxiliary inversion grammar rule encodes as part of its triggering pattern a possibly superfluous set of *Sally's* features, e.g., that *Sally* is a Name, Sally's Gender (female), and so forth. Given the generalization scheme, the presentation of an additional example sentence like *Will John kiss John* will eliminate via set intersection feature *Gender*, leaving only the tag *Name* on the triggering pattern for the grammar rule. Provided that the stock of features is finite (perhaps this can be guaranteed by limitations on inherent conceptual abilities) and that the supplied examples are consistent (the adult contributes, error-free, the regularities), then this method settles many of the encoding problems that will come up in the context of this thesis.

Note that because this technique has been designed to select bundles of features *across* a sequence of sentences, it cannot solve the additional problem of finding the correct triggering features to select *within* a single sentence. Consider for example Subject-Verb agreement in English: the Subject Noun Phrase must agree in Person, Number, and Gender with the Main Verb. How is Lparsifal to know (1) that this agreement must be observed; and (2) that it is these features, rather than, say, Gender, that do the work of encoding agreement?

Once again, one could resort to a stop-gap conservative strategy. If the number of features that can ever be attached to a single token is finite, then one way to proceed is to simply err on the side of over-labelling, attaching all possible features to both the Subject and Verb of an example sentence. Then regularities can be observed by comparisons across sentences: if an agreement feature is not observed -- say, a sentence occurs with a masculine-marked Subject and an feminine-marked Verb -- then that feature is struck from the surviving list of agreement features.

Such a method is almost certainly too simple-minded to be completely correct. If it is not carefully controlled it can lead to serious problems of over-generalization because certain sentences may not exhibit agreement paradigms that are observed elsewhere. For example, in English, Case Agreement is often dropped in *wh* questions (e.g., *Who did Sally kiss* instead of *Whom did Sally kiss*). But it is not dropped following prepositions, e.g., *I know of her* - **I know of she.* Therefore, the acquisition procedure should not drop Case Agreement for pronouns just because it is dropped for *wh* elements. To avoid this problem, the acquisition procedure has been deliberately designed to be conservative in

its generalization. Roughly, two sentences prompt a generalization only if they are one feature different (with respect to the parser state). So for example, *Who did Sally kiss* and *Whom did Sally kiss* would tell the acquisition procedure that the Case feature could be dropped for *wh* elements; however, since no positive examples exhibiting a lack of Case agreement would be encountered for *she/her*, Case agreement would *not* be dropped in this situation. For details, see Section 4 and

Chapter 3.[10]

---

10. Fortunately, because the set of potential features is finite (and perhaps not even very large), one could supplement (even replace) the guess-too-large-and-prune strategy with sheer brute-force enumeration; one could simply try all the possible combinations. Unfortunately, if the acquisition of semantically-grounded features for syntactic rules actually proceeds in a trial-and-error fashion, then the acquisition procedure in itself will not be of much theoretical (and computational) interest. It would however predict that there should be a sharp distinction in the kinds of errors observed in child acquisition, paralleling the syntactic rule-parameter setting/ trial-and-error distinction. On the one hand, the parameter setting framework implies a largely monotonic, error-free course of acquisition: the basic (and correct) settings for rules for sentence phrase structure are acquired early, and, once fixed, are never withdrawn. On the other hand, trial-and-error learning would leave the child open to outright blunders that should be easily detected. The target should be a system that makes errors in just those places where children make errors. As has been noted, children do not make gross errors in their hypothesis of base phrase structure rules, except where these are the result of faulty lexical categorizations (e.g., assuming that *it* is just like an ordinary Noun Phrase; see Baker [1979]); they do make errors with morphological patterns that are, in essence, memorized. But it also appears that the areas where children make errors are precisely those where there is abundant positive evidence, and even adult attempts at negative reinforcement.

A purely enumerative strategy should probably be supplemented by a theoretical structure of its own, for brute-force trial-and-error does not seem to account for certain inflectional errors made by children. See Chapter 3 for a simple formal induction model for these cases. For instance, Brown [1973 page 140] observes that (English language) children overgeneralize all verb inflections to inappropriate verb stems, save for one: the progressive *ing* is employed correctly on only "process" verbs (e.g., *hitting*), and never attached (incorrectly) to stative verbs such as *want*. The implication is that children develop quite early a distinction between the two classes of verbs. But this separation into verb classes cannot be directly expressed in the current model for the acquisition of syntactic features. A richer theory might exploit Lieber's recent work [1980] on the structure of the lexicon.
S. Pinker [1980] has also independently proposed a modified brute-force approach that attempts to deal with some of these problems.
The tight restrictions on the form of syntactic rule acquisition proposed in this thesis also makes several highly specific predictions as to the errors that can possibly be made during the course of acquisition. Since acquisition proceeds by altering the "atomic" components of a single parsing rule at a time, an error must be representable as some combination of elementary mistakes in a complete rule. Whether or not this prediction is supported by observational data in children is a matter still open to dispute: see Mayer, Errich, and Vallian [1978]

There is some evidence in the child language acquisition literature that would argue for this more groping, trial-and-error acquisition for a proper subset of feature predicates, though only in strictly limited contexts. First, in some special situations adults can be observed to at least attempt the correction of a narrow class of syntactic errors -- intriguingly, just the sort of errors that might require acquisition by brute-force methods. For example, it is a commonplace that adults will respond to a child's errors in Subject-Verb agreement or the over-generalized use of certain morphological markings with negative reinforcement of some sort. (e.g., *Don't say, "Mommy are hungry", say, "Mommy is hungry"; Don't "Mommy goed home", say....*) However, it is not at all apparent just what the impact of these mini-training sequences is. For the classic example of a mother's repetition having no apparent effect see McNeill [1966 page 69]; also quoted in Wexler and Culicover [1980 page 509]; or Braine [1971].

What is crucial to note is that the sort of syntactic error that is the target of these brave attempts is typically exactly those cases involving (language particular) flags for features such as Person, Tense, Gender, and the like. For example, in English we are told that the morpheme "s" is appended to verbs to mark the use of the third person singular case: *John likes Sally* is fine, but not *John like Sally*. Likewise, the association of a particular phonological shape with a particular morpheme is arbitrary (and language particular); there is no apparent way for the child to know that it is "s" that must be appended (rather than some other morpheme). (Actually, three allomorphs for the "s" must be acquired: /s -- z -- iz/. The microstructure of allomorph acquisition is not covered in this report. It presumably lies at the interface between phonological representation and segmentation capabilities, though it has obvious implications for the proper formulation of parsing rules.)

2.2.4 Errors in the Input Data, Resiliency, and Order Effects

There is yet another way in which the assumption of positive-only evidence is an idealization: The assumption of positive-only input abstracts away from the realities of normal discourse, scattered as it is with ungrammaticalities, false starts, and the like.[11] How is an acquisition procedure to deal with these realities? If it simply *assumes* that all input data it receives is well-formed, one might well wonder whether ill-formed strings could lead it astray. On the one hand, since the system must by definition *change* as new knowledge is acquired, it must somehow be responsive to input data. On the other hand, it must not change too readily, or else it will fall victim to over-generalizations prompted by the wrong kinds of data.

The proper response to a possibly incorrect input datum is (perhaps unfortunately) related to a much broader and more difficult question -- the recovery from errors in linguistic performance generally. But because theories of performance and error recovery are still quite primitive (in comparison with the rich theories of linguistic competence), our theories of the non-ideal speaker-hearer in a non-homogeneous speech commmunity are relatively impoverished.[12] In the absence of explicit theories of performance under noisy input, the current research offers two solutions: (1) Design an (idealized) model so that it changes its rules *conservatively* and *incrementally*. Such a system is at least plausibly resistant to certain errors in the input. And, since *most* input is still grammatical, one still has available an obvious sort of frequency criterion as an evidentiary basis for rule change -- say, by attaching a confidence factor to a rule (prompted by a certain sort of input data) only if the same rule has been hypothesized a certain number of times. (2) Ensure that rule hypotheses are widely scattered from one another, so that small deviations from the "correct" path do not lead the acquisition procedure astray.[13]

Note also that this sort of resiliency to error would be a desirable property in general for an acquisition

---

11. A significant percentage of utterances the child receives may be ill-formed -- see Wexler and Culicover 1980, page 77. However, as is so often the case in the child developmental literature, the evidence is most unclear. As cited in Brown [1973 page 387], Labov [1970 page 42] suggests that after removing obvious ellipses and false starts, only a few percent of utterances are ungrammatical. Brown's view [1973] is that adult speech to children is mostly well-formed. This does not quite answer the question, however, since it could well be that even though just a few utterances are ill-formed, these are the crucial examples that "trigger" major acquisition decisions. For example, one would not want to literally fix base phrase structure order on the basis of a single example, the first one heard, since that might be ungrammatical. Again, the obvious way out of this problem is to add some kind of "strength" requirement -- fix a rule only if its encountered ten, or a hundred, times. We will simply leave this question open here.

12. For recent work on parsers that attempt to deal with the problem of noisy and errorful input, see R.M. Weischedel, J. Black, *If the Parser Fails*; P. Hayes, G. Mouradian, *Flexible Parsing*, Proceedings of the 18th Annual Meeting for Computational Linguistics, 1980.

13. See Horning [1969] for a probabilistic approach to "noise" in the input data that is based on an enumerative approach different from the methods of this research. Gonzalez and Thomason [1978] provide a similar method, associating probabilities with rewrite rules (stochastic grammars). Olivier [1968] advances a stochastic approach to word segmentation. See Braine [1971] for a more informal discussion of the same topic.

system even if the data input were error-free. This is because we would like the system to attain the same final state of syntactic knowledge as *independently* of the order of data presentation as possible. The reason for aiming at an order-independent acquisition model is once again part of the general methodological slant of this research: We assume that the external environment is as weakly structured as possible, and then determine what constraints must be imposed on the acquisition program to ensure effective learning. Consequently one wants to avoid as much as possible any reliance on structure supplied by the external environment, including that implicit in the order of presentation of examples.

Here it is important to realize that there are two different conceptions of "order" that might be invoked to aid acquisition. The first refers to a deliberately designed "training sequence" of example presentations -- a device that normally presupposes some rich structure to the external world, usually in the form of a teacher. We might call this the extrinsic ordering of examples. In contrast, the structure of the learner might indirectly impose a filtering effect on the examples that are even considered germane for acquisition; we will call this intrinsic ordering. If a system is designed in such a way as to ignore complex examples that are beyond its embryonic parsing skills, then only examples that are "not too hard" provide the basis for new rules. Even if examples are presented in random order, the filtering effect imposed by the requirement of interpretability to some extent automatically labels perfectly grammatical examples as "noise" (unparsable at the system's current level of knowledge) and others as valid material for acquisition (fully or incrementally parsable). In the next section the way in which the parsability condition is enforced will be spelled out in more detail; briefly, the acquisition program cannot be recursively invoked. A full example showing the impact of this limit on what can be acquired in any one step of inference is given in the analysis of the acquisition of passive constructions, later on in this chapter. In brief, the constraint makes truncated passives ("Bill was kissed") easier to acquire, a fact that agrees with the actual order of acquisition of passive constructions.[14]

---

14. Intrinsic filtering has been often proposed in the developmental literature. For example, Brown's [1973] account of morpheme acquisition proposes that acquisition order is based primarily on the grammatical and semantic complexity of the morphemes involved -- relative to the internal state of the child -- and not on frequency effects observed by the external world. This distinction between extrinsic and intrinsic order effects parallels the relative contributions of distributional (observation-based) and representational (grammar-based) inferences in the acquisition process. The position of this research is that while it is true that distributional learning plays some role in acquisition, for the most part it is grammar that is central; observations make no sense (in fact, are literally uninterpretable) without a representational foundation on which to ground them.

Note that the input condition as stated was that "the program (child) uses grammatical sentences *as evidence for its hypotheses*". Thus, it does not claim that ungrammatical sentences are not interpreted by the child. Rather, we might assume that, just as is likely for adults, one tries to makes as much sense as possible out of an ungrammatical utterance, exploiting situational context if need be. This is in fact what happens in the analysis of passive constructions; see later in this chapter.

2.2.5 Formal Learnability and Negative Evidence

Using negative evidence is dangerous on yet a final ground. From mathematical results, it is known that positive and negative examples paired with the appropriate labels "grammatical" and "ungrammatical" enable one to learn *almost any* language (see Gold [1967]).[15] While this result might seem fortunate, implying that negative evidence would be a boon, it also implies the existence of an *informant* who is carefully guiding the learner through some reinforcement schedule. As mentioned above, the possibility of explicit reinforcement seems hardly likely in the case of human acquisition of syntactic knowledge. Children simply do not seem to receive correction of this sort.[16] Interestingly as well, the assumption of informant presentation in a sense places the burden of language acquisition implicitly on the adult, not the child, since in order to determine the next piece of data to present, the adult must somehow know the internal state of the child's grammar.[17]

In short then, the reliance upon positive-only evidence is a key part of the methodological strategy of this research. Its aim is to see what constraints must be postulated on a representation in order that acquisition be possible *even with* impoverished input data. Note that this viewpoint is in sharp contrast to most artificial intelligence models of learning. For example, Winston's concept learning program [1975] made essential use of negative examples as a powerful source of evidence for hypothesis formation. Perhaps the most important discovery of the current research is that in certain domains the limitation to positive-only evidence need not be debilitating. In fact, at least for the acquisition of syntactic knowledge studied in this report, quite the reverse is true. One can make considerable progress by discovering what sorts of constraints must take up the slack that negative evidence (supposedly) provided.

---

15. More precisely, any recursively enumerable set of recursive languages is "identifiable in the limit" from positive and negative examples. However, the result is abstract, and so sets a very weak (though important) upper bound on what can be expected from computational methods: there are many psychologically unsettling aspects to Gold's result that make it unsatisfactory as a "realistic" model of acquisition. For one thing, it relies on an enumeration of the class of languages (or grammars for those languages), as if the full details of each possibility were known in advance, rather than the construction of the proper grammar via the fixing of various parameters. As a result, the Gold procedure takes enormous amounts of time, approaching the correct identification only in the limit. Secondly, the approach permits the wholesale rejection or acceptance of grammars based upon single input examples, a fact not really compatible with what is known about human acquisition.

16. Brown [1973 pages 387-388]: "...judgment and correction are not a royal road to the child's grammatical knowledge." See also Wexler and Culicover [1980].

17. As pointed out by Newport, Gleitman, and Gleitman [1977]; see also Wexler and Culicover [1980 pages 69 and 75]. But note that children apparently do receive negative reinforcement for *semantic* well-formedness -- the adult says "Huh?" in response to a meaningless string. As Anderson [1977] shows, this *semantic* supplement can add (in principle) sufficient information to establish learnability from just *syntactically* correct sentences.

For further discussion of research in mathematical theories of language learning and their relevance, see Appendix One of this report, or the review article by Pinker [1979].

2.2.6 Semantic, Pragmatic, and Contextual Information

In addition to a surface string already divided into words, there are other sorts of input data that have traditionally been assumed helpful to a hypothetical child acquiring syntactic knowledge, and therefore of possible importance for an acquisition program. Foremost among these is what is usually dubbed "semantic knowledge" or sometimes, "contextual" or "world knowledge" -- all the information that might contribute to one's understanding of an utterance besides the properties of the utterance's surface string. The way such information could in principle be of assistance to the learner is clear: if syntactic knowledge involves the mapping of surface strings to mental representations (including perhaps what some might loosely call "meaning"), then at some level the acquisition of syntactic knowledge means the acquisition of the correct function to carry out this mapping. Contextual information could help because in certain cases it might independently specify (part of) the underlying representation, pinning down one end of the mapping.

It is worthwhile to further examine the assumptions implicit in this approach, for this will help tell us whether the assistance in principle can also be assistance in fact. It is the general assumption of many linguists that the underlying representation must, at a minimum, perspicuously encode the predicate-argument structure and thematic relations of the sentence -- that is, an identification of the main Action(s) of the sentence, the Agents of those actions, Objects of those actions, and so forth. Roughly, who (or what) does what to whom must be easily recoverable from some representation built up during the analysis of the surface string; this informal statement of the problem will suffice for our purposes here.[18] Consider a logical decomposition of the problem, as indicated in Figure 2.7 below. The recovery of "semantic information" from the surface string has been broken down into the composition of two functions, P and PA. The first function, P, maps the surface string into a rather abstract annotated surface structure (a parse tree, plus additional information); the output of step 1 serves as the input to the function PA, that actually recovers the thematic relationships of the sentence. The parsing function P is designed to make PA's job trivial. That is, the annotated surface structure is designed precisely to encode (via its hierarchical tree structure) the major predicates of a sentence, their arguments and scoping relations to other elements, and, via a default association of argument positions of predicates with thematic roles, the corresponding thematic relations. In one current theory of transformational grammar, the Government-Binding theory (Chomsky [1981]), this demand is honored by insisting that the abstract surface structure of sentences be "semantically transparent," in the sense that the *form* of the augmented labelled bracketing must reflect the predicate-argument structure of lexical items. This demand is called the "Projection Principle," since it amounts to the

---

18. But this is not all we might reasonably demand of the underlying representation: it might also include quantifier scoping, topic and focus; or other relationships. The identification of arguments to a verb, though, is plainly required in all these cases as well.

restriction that the predicate argument structure of lexical items be "projected" through all linguistic levels of representation.

The Projection Principle was, in fact, implicitly adopted by Marcus. For instance, when the Marcus parser encountered the verb *kiss*, it entered a state that expected the appearance of an NP immediately to the right of *kiss*. (In the actual implementation, this was handled by entering a packet reserved for verbs that took an NP Object, such as *kiss*.) Thus the argument structure of *kiss* was assumed to hold at the level of labelled bracketing, the level of structure the parser attempts to build. Actually, Marcus' implicit use of the Projection Principle went one step beyond a representational claim. Marcus also adopted a stricter computational analogue of the Projection principle. Not only is argument structure projected to the surface, but this structure was assumed to be recoverable on-the-fly, that is, as the construction of the parse tree proceeded left-to-right through the input string.

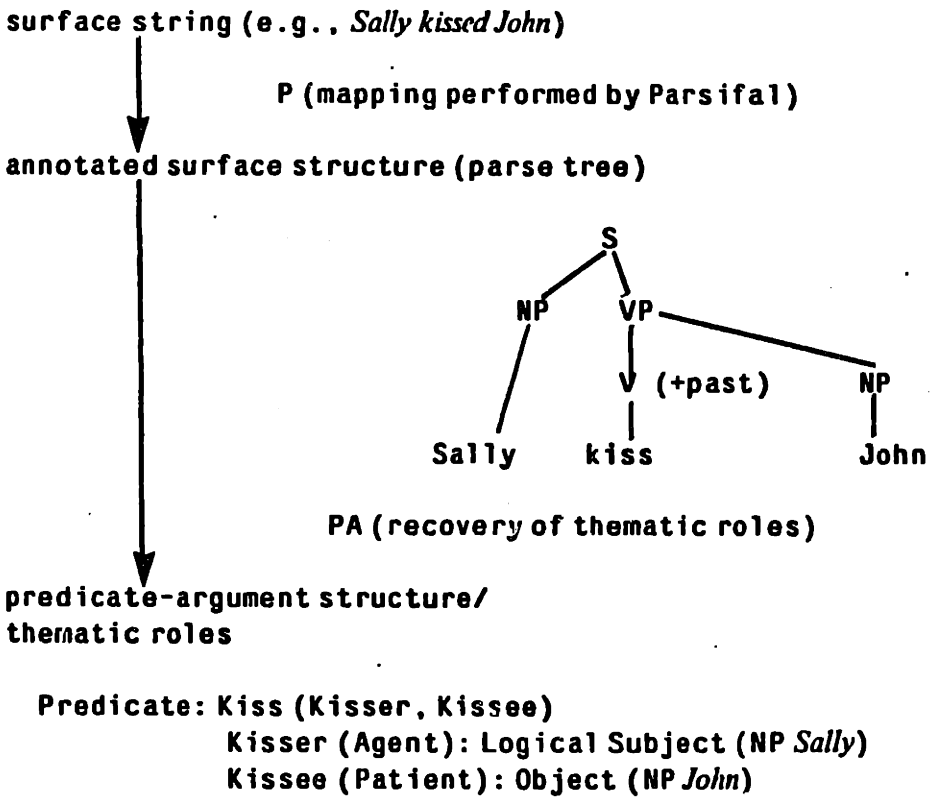


Figure 2.7 - Recovery of predicate structure as the composition of mappings.

What is the role of extra-syntactic information in acquisition? Suppose we supply contextual,

pragmatic, or thematic information sufficient for the learner to recover the Predicate-argument structure as listed at the bottom of Figure 2.7 -- (perhaps without even referring to the utterance itself). Then, if it were the case that (1) PA^{-1} were known (that is, we knew the inverse mapping from predicate structure to annotated surface structure) or (2) the annotated surface structure was isomorphic to the predicate-argument structure then the contextual information would help. For example, if it is not known that "kiss" requires (as a verb) a Noun Phrase Object argument, then the extra-syntactic recovery of this fact via predicate argument structure is a tremendous boon. (If "kiss" is already known, then we assume that its argument requirements are known.) In short, extra-syntactic information can be used to learn new words.

Not surprisingly, what this analysis has revealed is that extra-syntactic information is most easily used precisely when supposed "semantic" forms are close to syntactic forms, and all the more so when syntactic forms are isomorphic to literal surface strings of words. This close correspondence between surface and "semantic" form typically holds only in the earliest stages of acquisition, when (as often observed) there is but a single surface form for every underlying semantic structure. But this is simply the observation that the Projection Principle holds in early child language. In fact, one common strategy of children at this early stage is to simply "read off" the thematic roles of Agent, Action, and Object in a fixed sequence directly from the surface order of words (Fodor, Bever, and Garrett [1974]). There is simply no need for any other "strategy" because the structural relationships among constituents is reflected in the linear ordering of words themselves.

But what happens when surface form and meaning part ways? The observations above suggest that any *large* deviations between surface form and known syntactic/semantic relationships is likely to render an utterance simply uninterpretable.[19] Suppose, however, that semantic and syntactic form never "part ways," because syntactic form is augmented with abstract structure so that it always reflects underlying predicate-argument structure. Roughly speaking, suppose that sentences "wear their meanings on their sleeves." This constraint -- amounting to the Projection Principle -- would seem to be *prima facie* the right assumption to make, and it is the one that will be adopted here.

The adoption of the Projection Principle still leaves us with the problem of how the assignment of arguments to structural positions is made initially. For instance, what tells the child that *ball* in *John kicked the ball* is the Goal or Patient rather than the Agent of the sentence? Here it appears that there is but one approach that has been suggested in the literature, though in various forms (for example, by Anderson [1977]; Wexler and Culicover [1980]). This is to assume that, to some degree or another, the acquisition procedure can recover the correct assignment of thematic roles to verbal arguments independently of the usual linguistic rules used to recover thematic role assignments (the mappings P

---

19. See Limber [1973 page 172] for an earlier exposition of this position.

and PA). So for instance, in a sentence such as, *John kicked the ball*, it is assumed that the acquisition procedure knows that *John* is the Agent of the sentence, and has had some effect in some way or another on the Goal or Recipient, the *ball*. In Anderson's work, this assumption is explicitly made in the form of providing the acquisition procedure with a "semantic network representation" of the input sentence.[20] Wexler and Culicover do not assume that thematic structure is explicitly furnished, but they do assume that the learner is provided with a base phrase marker (a deep structure representation) of each surface string. Assuming that thematic roles are "read off" a deep structure tree, this is actually a stronger condition than simply positing that correct thematic role assignments can be recovered. It is a stronger condition because it assumes that structural relationships among thematic roles are also independently recoverable. Wexler and Culicover's theory is set within an *Aspects*-style transformational theory, where deep structures are used for semantic interpretation. A base phrase structure represents canonical predicate-argument structure, and so satisfies (trivially) the Projection Principle. Fixed positions in the deep structure tree are associated with certain grammatical relations, e.g., the NP under S is the Subject, the first NP under the VP is the Direct Object, and so forth.[21]

How exactly could thematic role assignments be determined without recourse to linguistic knowledge? It has often been imagined that an initial association of linguistic tokens with this quasi-pictorial (and Wittgensteinian) semantics is carried out literally by some kind of interaction with the real world -- either visual observation, or participation in the actual act involved. [22] However, since such proposals are not very precise, we shall just leave this question as an unresolved problem, and simply assume that the association between certain linguistic representations (roughly, Noun Phrases) and thematic roles (such as Agent, Patient, Goal, Instrument) can be made. Note that several researchers (such as Keil [1979]) have observed that concepts such as Agency, Instrumentality, and the like, are acquired quite early by children, as seems reasonable. Thus it would be at least natural for children to be able to assign these characteristics to objects in the world (however that is done).

It should be stressed that this approach does <u>not</u> necessarily imply that the predicate-argument structure must *always* be independently recoverable from extra-syntactic context alone in order for acquisition to proceed. In many cases the proper assignment of (for example) thematic roles to the constituent Noun Phrases in a paticular sentence is <u>not</u> a prerequisite for the formulation of parsing

---

20. In fact, Anderson assumes that this semantic network is a minor structural deformation of a labelled bracketing of the surface string.

21. However, it should be noted that Wexler and Culicover realize that the assumption that a complete base structure is furnished with every example string is a strong condition that might perhaps be dispensed with. In fact, they propose that base structures could be inferred directly from thematic assignments plus conditions on properly formed base phrase structures rules, roughly, X-bar theory. This proposal is quite close to the approach adopted in this thesis.

22. Indeed, there is some evidence that " active participation" in some sense is required: e.g., volitional vs. non-volitionally practiced motor skills.

rules. This is because the absence of a needed rule of parsing can often be signalled by the detection of a missing item in the input string of words itself -- a "gap" -- and this can sometimes be done no matter what the thematic role assignment of the missing material.

To illustrate, consider the relationship between active and passive sentences in English. In most cases, the English passive apparently interchanges the canonical positions of the "doer" (Agent) of an action and the thing affected by the action (the so-called *Goal* or *Patient*):

```
active surface form:      Bill kissed  Sally
                           |           |
thematic roles:          Agent      Affected Object

passive surface form:     Sally  was kissed by  Bill
                           |                    |
thematic roles:        Affected Object         Agent
```

Part of the (English) language learner's job is to figure out the syntactic cues that English uses to flag such differences (so that parsing can proceed in the absence of confirming situational context information or so that generation can conform to the syntactic rules of English). Consider the truncated form of the passive above, *Sally was kissed.* At the minimum, what is *syntactically* significant for parsing here can be broken down into three steps: (1) parsing *Bill* as a Noun Phrase, as usual, and attaching it as the surface Subject of the parse tree; (2) discovering the presence of a gap after the verb *kissed*, corresponding to the obligatory second argument that appears in the "canonical" form, e.g., *Bill kissed Sally*; and (3) noting that the item *Sally* appearing in the canonical agent position is actually the "object" of the action. At least for this simple example, the first two steps do not seem to require any special attention from situational context at all; it is simply a matter using rules already at one's command to analyze the leading Noun Phrase. Then, all that one needs to know is that *kiss* requires two arguments and that there is no second argument present in the input string. But this is simply to say that one knows how to properly assign thematic roles in the sentence, *Bill kissed Sally.* As for step three, although the linking of *Sally* to its "proper" canonical argument position would at first seem to imply the direct use of situational context, this is not strictly necessary. Rather, one might assume a uniform rule of interpretation, operating in a purely mechanical way, that links the Noun Phrase *Sally* to the already-marked gap. This is in fact the approach taken by Chomsky in *On Binding* [1980].[23]

---

23. Chomsky has deliberately tailored the binding relationship between gap and preceding Noun Phrase after other anaphoric phenomena, e.g., the connection between pronouns and their antecedents.

One final point is that one must ensure that *Sally* in the usual Subject position is not interpreted as an Agent. Observe that *Sally* is now linked to the Object position, and as such is interpreted as the Patient or Affected Object of *kissed*'s predicate-argument structure. Suppose that this linking means that *Sally* cannot fill another argument position for *kiss*, playing the dual role of both Agent and Affected Object. Then the Subject position in this case must simply be missing. In fact, there will be other positive examples that illustrate this situation, e.g., a predicate adjective form such as, *John was sick*. Here, there is no explicit Agent; rather, *John* serves as the Affected Object of the sentence, which means something roughly like, Sick(John). Thus there will be positive evidence for an Agent-less *be*-predicate form that can inform an acquisition procedure that Subject position is not necessarily associated with Agency. (One might even take this deduction a step further. Suppose we simply *define* a new thematic role, consisting of the situation where an NP argument to a verb is assigned the role of Affected Object -- something like "predication", in the sense of Williams [1981a] -- and yet that NP appears in surface subject position. This role could simply be called the theme of the sentence. This example indicates that syntactic predicate-argument structure in conjunction with the Projection Principle could be used to infer new thematic roles, rather than, as is usually assumed, the other way around.)

Note that this method still leaves open the possibility of a "semantic" or situational check on the resulting linkage (one could refer to the thematic role assignment derived from situational context to make sure that the syntactic machinery had operated correctly), but the need for this is far from clear in the truncated passive example. For given this account of the truncated passive,[24] acquisition of the full passive analysis (*Sally was kissed by Bill*) may not be so mysterious. If the gap after *kissed* can still be assumed detectable, then its binding to *Sally* can be effected by exactly the same rule as in the truncated passive. However, some additional interpretive rule is required to determine that the agentive *by* phrase serves as the first argument to *kiss*.

Summarizing, the thematic role assignment machinery is intended to be used sparingly (in the best case not at all), and, even when it is used, it is to be employed only as a check for well-formedness. One way that the check might be used in the acquisition model is that if the structure resulting from the analysis of a sentence is ill-formed (i.e., produces a mis-assignment of predicate-argument relations), then any new syntactic rules acquired during the processing of that sentence could be

---

24. Incidentally, the truncated passive appears before the full passive in children's speech (Limber [1973]) -- as predicted by this account. See Section 5 of this chapter for a complete account. See also Maratsos [1978] for additional evidence, and Weinberg [1982] for a more detailed linguistic analysis.

thrown out.[25] In brief, the more that uncertain cognitive machinery is invoked as a key part of the explanation of acquisition, the more we draw away from a precise and testable theory. And matters become rapidly more serious when one turns to developmental issues. If little is known about the precise representation of thought in the mind, still less can be certain about its development; a model that attributes adult representational capabilities to an embryonic device has a good chance of being wrong.

Given these pitfalls, it would seem appropriate to make as few assumptions as possible about the additional cognitive machinery that may be appealed to in the acquisition model -- and better still if it can be dispensed with altogether. The primitive conceptual basis that is invoked should be at least one that can plausibly be assumed available to a developing child.

Because Lparsifal is based upon Marcus' work, it has all of Parsifal's semantic apparatus at its disposal for checking its acquisition in the manner described above. This includes:

> (1) A basic stock of *thematic role* primitives, consisting of, e.g., *Agent, Goal* or *Affected Object, Patient, Recipient.*
>
> (2) The correct assignment of constituents to thematic roles. Such sentences correspond to positive example utterances that appear in a context rich enough for a hypothetical learner to draw the proper connection between thematic roles and sentence constituents. For example, given the sentence *Sally was kissed by Bill*, the program has available the knowledge that *Sally* is the *Affected Object*, the Argument to the Predicate *kiss*; Bill is the logical Subject of the Proposition (the full Sentence).
>
> (3) By default, any arguments to a verb are assumed to be obligatory until proven otherwise. This rule follows the Subset Principle for acquisition from positive-only evidence (see Chapter 3 for discussion).[26]

The actual input to the acquisition procedure that the user must supply looks like this:

---

25. There are several other reasons for exercising caution in the provision of extra-syntactic information. First, one risks begging the hard questions about acquisition: if the supplied predicate-argument representation encodes the hierarchical relationships among all the constituents of a sentence, then we have just assumed the syntactic knowledge that was to be acquired. Second, there are unsettled questions about just what the canonical underlying representation should look like. If it is known but in rough (probably incorrect) outline, then one would seem to lose the ability to tie down one end of the map from surface string to underlying representation. Finally, the ability to construct a canonical cognitive representation from context alone, if taken at psychological face value, would imply that all people across all languages assign thematic relations such as Agent and Patient in the same way. This point has been made by Wexler and Culicover, [1980 page 406].

26. T. Roeper [1981] has also suggested a default obligatory requirement for arguments to predicates, based on the same learnability motivation.

```
Sentence:
   I gave a book to Bill.

Supplied Predicate-Argument Structure:
   Predicate: give
   Arguments: book, Bill
   Logical Subject of Proposition: I
```

It should be stressed that this information is only potentially available to Lparsifal; the currently-implemented procedure does access the argument structure of predicates, but need not make use of any other semantic checks while the parse is in progress.

2.3 Component 3: The Acquisition Procedure

Having briefly sketched the knowledge to be acquired, the initial state of knowledge, and the input data to be used for the construction of new rules, it remains to outline the acquisition procedure itself. Earlier in Chapter 1, Figure 1.1 depicted the course of acquisition as the development of a sequence of parsers. The initial parser P_0 has no grammar rules and just a single skeleton base rule. A sequence of grammatical example sentences is then given to the acquisition procedure, which proceeds according to the following steps. (Some details will be added to this procedure in this next section after we discuss more fully the way in which the modified Marcus parser works.)

> Step 1: Attempt to parse the sentence left-to-right in a single pass, using currently known base rules and grammar rules and the Marcus Parsifal machinery. If there are no failures while processing the sentence (the parser is never blocked because there are no grammar rules that can trigger) and afterwards (the output is a complete parse tree with a valid thematic role assignment), read in and attempt to parse the next sentence. If the parse fails at any point during the analysis (no currently known grammar rules trigger), attempt acquisition.
>
> Step 2: Enter acquisition phase. Note Instantaneous Description (ID) of the parser at the point of failure (current active node, cyclic node immediately above the current active node, contents of the input buffer, and name of the currently active packet/base rule). The left-hand context of the ID consists simply the two nodes of the active node stack (including any annotations of these nodes prompted by rule attachments, see below) plus the name of the currently active packet (or base rule context). The left-hand context summarizes all the information about the parse required by the interpreter to make its next parsing decision; see Chapter 5 for additional discussion. The right-context of the parse consists simply of the

contents of the buffer. (This definition of an instantaneous description will be modified somewhat after it is shown how the packet system can be eliminated. The term "packet" will be replaced with "X context.")

Step 3: Attempt to build a single new (base phrase structure) rule corresponding to the packet (X context) currently active. If successful, store the new rule as an instantiated X-bar template (via the parser action *attach*). Percolate the features of the attached item to the Head of the phrase under construction. Prompt the user for a mnemonic name for the new rule. The *pattern* of the new rule is simply the ID of the parser. In effect, we have performed a *reduction*, performing the expansion A-->a backwards. Annotate the currently active node (top of the active node stack) with the name of the rule so formed. Continue with parse. If the *attach* is unsuccessful, (the new rule violates constraints of the X-bar theory for the current X context) and if the current portion of the phrase structure expansion under consideration is optional, advance to consider the next portion of the X-bar expansion.

If this particular phrase is completed, then drop the entire X-phrase into the buffer. Completion is signalled if either (i) known arguments have been attached and the next item in the input buffer is the end-of-sentence marker or an item that cannot be attached to the current phrase; or (ii) predicate-argument knowledge indicates that the phrase is complete.

Otherwise, go on to Step 4, and attempt to build a new, non-base grammar rule.

Step 4: Attempt to build a single new grammar rule. Select as the new pattern of the rule the Instantaneous Description noted in Step 2, and as its new action one of the three actions (*switch*, *insert-lexical item*, or *insert-trace*) (The possible grammar rule actions and how they are chosen is discussed further in the next section.) The reason for ordering the actions in this way is discussed immediately below. If successful, update the rule database, first checking whether the new grammar rule performs the same action as an already-known grammar rule (i.e., performs the same operation at the same parse tree location). If so, generalize the grammar rule via one of two conditions (see immediately below for a discussion of rule generalization). Either (1) form an equivalence class of items corresponding to the item just attached; or else (2) split an existing class; or (3) write a generalized grammar rule. Prompt the user for a mnemonic name for the new rule. Continue with current parse. If unsuccessful with all three candidate actions, and the X component under construction is optional, drop the phrase under construction into the buffer and continue; otherwise, stop the parse of the current sentence and analyze the next sentence. If the sentence as constructed violates known the known predication representation, then remove any newly created rules from the

rule database.

Several properties of this procedure are worthy of note.

★ **Process model realization.**
At each step in the acquisition sequence, the model always has its knowledge of syntax available in the form of a working parser -- albeit an incomplete one. Thus the procedure provides a functionally realized representation of syntactic knowledge of ever-increasing sophistication.

★ **Constructive and incremental acquisition.**
(*"One error-one rule."*)
The system adds only a single new rule at a time to its knowledge base, rather than hypothesizing and rejecting whole sets of rules (or grammars) at a time. Thus the procedure does not proceed by *enumerating* the sets of possible parsing rules (or grammars), and testing each for adequacy in turn, the method that was used by many previous acquisition programs for syntactic rules (see Feldman [1972]; Horning [1969, 1971]; Biermann and Feldman [1972]; Fu and Booth [1975]; review in Pinker [1979]). Rather, the program is *constructive*, building the right parser by filling in the details of an initially-provided set of base rule templates and assembling one at a time a set of pattern-action grammar rules. As will be shown, this incrementality condition (working in conjunction with a restriction on recursive calls to the acquisition procedure) has the effect of imposing an intrinsic order to the sequence in which rules are acquired.[27]

★ **No recursive entry into the acquisition procedure.**
The procedure cannot call upon itself to construct a new rule while already in the midst of constructing a rule. This stipulation ensures that only one new rule can be constructed for each detection of a missing rule ("one error-one rule"). If recursion is necessary, the current acquisition attempt is abandoned. In part, this stipulation is justified by the principle of finite

---

27. Note that incremental acquisition may well prove to be far too strong a theory in the case of child acquisition: if there can be dramatic "radical reorganizations" of grammars -- for example, the abandonment of one whole set of phrase structure rules for another then the model proposed here will not suffice. To take another example, consider the case of German. If simple German declarative sentences are all in S-V-O order (perhaps the result of a "Verb Second" rule), then an acquisition procedure hearing only S-V-O sentences must be driven to the conclusion that German is a Head-Complement language. But then it will later hear sentences in embedded clauses where the order is S-O-V, and so must abandon its previous conclusion. This is only an apparent contradiction, however. Thiersch [personal communication] has observed that in fact most simple German sentences used in conversation are Verb final.

On the other hand, incremental decisions, such as the decision that Heads follow Complements, can have significant impact on the appearance of surface strings. It follows therefore that an incremental acquisition model does not necessarily rule out the possibility of apparent radical reconstruction.

error detectability: if an error is detectable, it should be detectable in a "local" radius about a failure point. That the finite error condition is a proper characterization of the class of (humanly) learnable grammars is the centerpiece of the formal learnability work of Wexler *et. al.* (Hamburger and Wexler [1975]; Wexler and Culicover [1980]; for further discussion, see Chapter Five.

★ **Conservative acquisition.**
Each new parser P_i is determined by the *current* input example sentence plus the previous parser, P_{i-1}. This implies that the acquisition procedure does not store previous example sentences it has encountered (though it does so indirectly via the knowledge in its rule base). Conclusions about the structure of a new rule are made by drawing upon knowledge of past rules, the current example sentence, and specific constraints about the form of all rules. It is this reliance on past knowledge that provides an incremental, conservative basis to the system's development. See Chapter 3 for a more formal discussion of incremental acquisition.[28]

★ **Base rules acquired before grammar rules.**
The details of basic phrase structure rules are set before more particular grammar rules are acquired. For instance, the system would assume that the basic word order of a sentence is generated via phrase structure rules, for instance, Sentence⇒ Noun Phrase-Verb Phrase; Verb Phrase⇒ Verb-Noun Phrase, before presuming that the ordering was generated via the movement specified by some grammar rule. The reason for this ordering follows a principle of acquisition from positive-only evidence, discussed below.

★ **Lexical acquisition as equivalence class formation.**
Items that behave alike with respect to parsing, e.g., the class of modals (*should, will, can...*) are placed in the same category.

Four points about the acquisition model deserve further discussion at this point: (1) Rule generalization; (2) The order in which rule actions are attempted; (3) Annotation of the active node with the names of rules that build that node; and (4) The order in which rules are executed. Let us cover each of these in turn.

(1) Rule Generalization and the Acquisition Procedure

---

28. Thus the acquisition procedure is intensional, in the sense of Chomsky, 1975, pages 119-122: the current parser plus the new input datum (grammar in Chomsky's discussion) fixes the next parser. For a discussion, see Wexler and Culicover, pages 93-94.

As mentioned, there are basically two situations where rule generalization can occur. First, new lexical categories can be formed via the collapse of terminal items into equivalence classes. This mode of operation is appropriate whenever the *action* of the new rule just acquired is *attach*. This is because the *attach* rule is simply the reverse of the expansion, $A \rightarrow \alpha / \Phi \_\_ \psi$, that is, α is being reduced to A in the context Φ on the left and ψ on the right. (See Chapter 3 for additional formal discussion of left and right context). Equivalence class formation is based on the notion of "state" from automata theory. Recall that the left context and right context forms the ID (instantaneous description of the parser). The state of the parse is simply the left-context, plus the node being reduced. Consider now two parser states q_i and q_j, and the succeeding states of the parser prompted by the analysis of the input tokens *a* or *b* just attached to the parse tree, q_{i+1} and q_{j+1}. If the two initial states and the two succeeding states are equivalent ($q_i = q_j$ and $q_{i+1} = q_{j+1}$) then, since the parser is deterministic, *a* must be in the same equivalence class as *b*. For example, consider the sentences, *John could kiss Mary*; *John will kiss Mary*. *Could* and *will* are followed by the same strings (hence must lead the parser through the same sequence of states, leading to acceptance, at least given this example); the parse up to the point where *could* and *will* are encountered is also identical in both cases, so the parse state at the point where each is attached to the parse tree must be the same. Thus the conditions for equivalence are fulfilled, and *could* and *will* should be placed in the same (lexical) equivalence class. Conventionally, this set is called the class of *Modals*.

There is a second case where generalization should occur. If the left-hand context is the same for two parses, and the input token attached is also identical, then the succeeding state must be the same in both cases, that is, $q_{i+1} = q_{j+1}$. But the succeeding state is represented by the items of the right-hand context, i.e., the current input buffer items. This right-hand context represents the same state only if the corresponding elements in the buffer are members of the same equivalence classes. Therefore, the corresponding next tokens in the input buffer must be members of the same equivalence class. For instance, consider the following sentence pairs: *John could take the book*, *John could buy the book*. The parser state after attaching *could* must be the same in both cases; therefore, *buy* and *take* are members of the same equivalence class. We record this fact by generalizing the grammar rule that attaches *could*:

buffer 1: [could][take, Verb, tenseless][the]
buffer 2: [could][buy, Verb, tenseless][the] --> intersect features

generalized
rule: [could][Verb, tenseless][the]

Thus, rule generalization by merger occurs when it is known for certain that the buffer will lead to an equivalent parser state. Note that this second kind of generalization is actually a variant of the first: *take* and *buy* will be placed in th same equivalence classes by the attachment that occurs after *could* has been attached to the parse tree, just as *will* and *should* are merged. As Chapter 3 describes in more

detail, this method is simply a variant of a so-called "clustering" or "k-tail" approach to finite-state induction (if the buffer holds only single tokens). What this means is simply that two tokens are declared to be in the same equivalence class (hence are denoted by identical non-terminal labels in the acquired grammars) just in case their trailing k-cell suffixes, or "k-tails" are identical. If the buffer can hold complete constituents, or sub-trees, as in the Marcus parser, then this method is actually induces a restricted class of tree automata. This formalization allows one to calculate bounds on the running time and correctness of the acquistion procedure; see Chapter 3 for details, and a full application to the induction of the Auxiliary system of English.

Later examples may force the *splitting* of a formerly merged class. For example, consider the examples, *I did go* and *I will go*. This example forces *did* and *will* into the same class. However, a later example, such as, *I will have gone*, along with the *absence* of the example, *I did have gone*, shows that this merged class must be split. For additional examples, see Section 2.5 and Chapter 3's discussion of the induction of the Auxiliary system.

For the other grammar rule actions, *switch* and *drop trace*, no token is attached, so rule generalization must work somewhat differently. The basic equivalence class approach is still used, however. The method is as follows. If the trailing suffix as stored in the current buffer is identical for two rules except for the first buffer element, and if the left-hand contexts of both rule patterns are the same (so, in particular, both rules must be associated with the same X context), then the rules are merged to yield a new, generalized rule by forming the intersection of the features in the first buffer cell. For instance, consider two instantiations of an Auxiliary inversion rule:

Rule 1: [Did -N +V -Arg][Sally, NP][kiss, +V, +Arg]
Rule 2: [Could -N +V -Arg][Sally, NP][kiss, +V, +Arg]

Here, the "-N" feature simply stands for the fact that *did* and *could* are known to be *not* Nouns -- because they do not denote objects in the world. The -Arg feature means that *did* and *could* do not take NP arguments -- they do not assign thematic roles to Object NPs as, e.g., *kiss* does. Because the trailing buffer suffixes are the same in both cases, a merged, generalized rule is formed:

Rule 3: [-N +Tense -Complement][NP][+V -tense]

It is not obvious why these rule generalization procedures should work. Indeed, the claim that class merger can be determined just by inspecting the buffer and the local left-hand context of the parse is a strong one; it plays the same role here as the notion of Finite Error Detectability does in the Wexler and Culicover work. The claim is that a 3-constituent buffer plus two-cell left-hand context actually

suffices to detect all cases of non-equivalent states.[29] This constraint is intimately connected to Wexler and Culicover's demand that all possible transformational errors be locally detectable on *some* phrase marker. In fact, it can be shown that Marcus' constraint that sentences be locally parsable deterministically is actually stronger than this demand, in that Marcus parsability implies Wexler-Culicover learnability, but not conversely. Again, for details, see Chapter 3 and Chapter 5.

(2) Rule action ordering by the acquisition procedure

Above it was stated that the acquisition procedure attempts to find a single new action by trying the following sequence:

1. Attach
2. Switch
3. Insert specified lexical item (e.g., *you, there*)
4. Drop trace.

Why is this ordering of actions attempted and not some other? This order has been designed to obey the Subset Principle, introduced in Chapter 1 and described in detail in Chapter 3. Briefly, this ordering of hypotheses is designed to generate as small a class of languages (surface strings) as possible. The tightest assumption possible is that no movement is permitted; i.e., that all surface strings are base generated unless evidence is provided to the contrary.

From another point of view, however, it seems unnecessary to invoke any ordering at all, because the possible action seems to be in nearly complementary distribution. Consider first what it must mean to be forced to order the *drop trace* and *attach* actions. This implies that there must be some case where if the order of actions was (1) try drop trace and then (2) try attach, then an incorrect rule would be acquired. But this must mean that the *drop trace* rule passes the required test for rule acquisition, yet is incorrect, and the *attach* rule passes the required X test for successful acquisition as well, and is correct. Otherwise, the order 1- try *attach* and then 2 - try *drop trace* would result in a failure of the *attach* to apply, and the *drop trace* action would be attempted, just as if the order were the other way around. Can we find a case that meets this condition? The success of a drop trace rule means that a trace is dropped, and then, in the very next attach action, attached to the active node stack. When is this possible? First of all, NP traces must be at least governed.[30] Since government subsumes the

---

29. It is not clear whether the three-cell limit is psycholinguistically significant or not. Given the assumption that an "S" consists of NP INFL VP, then the input buffer can hold an entire S. In addition, grammar rule patterns can access two nodes of the active node stack, the current active node and one cyclic (S or NP) node above the current active node. At most then a grammar rule can access an S as the cyclic node, an S as the current active node, and the constituents NP INFL VP in the input buffer – roughly a two-sentence range, the current S being parsed plus the S dominating the current S.

30. Actually, traces must be *properly* governed, a refinement of government.

notion of subcategorization, then an attachment is certainly permitted if the NP is subcategorized, as in, *Bill was kissed -- NP trace*, where *kiss* requires an NP Object. Could a normal *attach* action work in this case? Apparently not; this would amount to a sentence such as, *It was kissed Bill.* In the Government-Binding Theory [Chomsky, 1981] this fact is accounted for by the quasi-adjectival nature of *kissed*; apparently, *kissed* does not assign (Objective) case to its subcategorized NP. Since lexical NPs seem to require case in order to be "visible" to phonological processes[31] a lexical NP cannot appear in a non-case marked position. But an NP trace, being phonologically unrealized, can appear in this position. Finally, the antecedent of an NP trace must be in an athematic position, such as Subject position in such sentences as, *It seems that John is asleep*, and must meet the Subjacency condition. This last requirement is automatically fulfilled by the operation of the Marcus parser, as modified in this research. (Rule patterns can only access the current active node, and at most one cyclic node beyond that.)

If this analysis is correct, then the drop trace and Attach actions can never conflict -- assuming that it is known that *kissed* does not assign Objective Case to the NP it subcategorizes for. How could an acquisition procedure know this? There are simple positive examples where the choice between *attach* and *drop trace* never arises, because there is no alternative lexical NP to attach. For example, consider truncated passives, such as, *John was kissed.* Here, only the action *drop trace* can possible succeed. But this example indicates exactly what the procedure needs to know: that the quasi-adjectival form *was kissed* "absorbs" Case. See later in this chapter for additional discussion of this example. Assuming that such "clear" cases are acquired first -- as seems to be the case (see Weinberg [1982]) -- then no problems arise.

What about the relative ordering of attach and switch? The switch action is constrained to operate in the following situation: Nodes A and B are immediately dominated by some node X. The order A--B is *not* a possible base phrase structure order, but the order B--A is. If *attach* fails and *switch* succeeds, then this means that both the order A--B and B--A can appear in surface string form, one as a base structure, one as an inverted structure. (E.g., *John has been kissed -- Has John been kissed.*) If both *attach* and *switch* fail, then neither A--B nor B--A are base structures. Conversely, if both actions succeed, then both A--B and B--A must be base structures. Finally, if *attach* succeeds and *switch* fails, then B--A is a base structure, but not A--B, and A--B does not appear as a surface string. Given these conditions, does the order (1) try attach then (2) try switch matter?

(3) Annotation of the Active Node

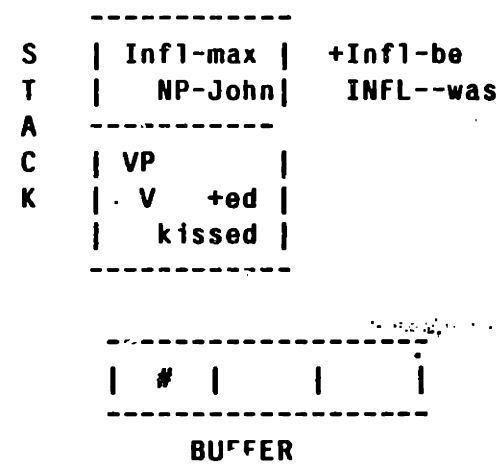
As described above, after a grammar rule executes we label the currently active node (the node on top of the active node stack) with the name of the rule just executed. (In the case of a newly acquired rule,

---

31. Vergnaud's case filter; extended by Aoun [1979] as a general "visibility hypothesis."

the name is a uniquely generated name.) This is done for two reasons: (1) to mark the current left-hand context of a parse, so that certain rules will execute properly and (2) to provide an annotation for a (hypothetical) semantic translation routine to distinguish between sentence variants, e.g., declarative and auxiliary-verb inverted forms.[32] The name of the rule is simply attached as one of the features of the currently active node.

As an example of the first use of annotation, consider a rule for a passive construction, such as, *John was kissed.* When the parser reaches the point where it is analying the complement of the verb *kissed*, a passive rule triggers. Note that at this point the currently active node is the VP:



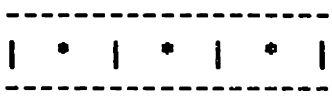
(# is the end of input marker)

The basic action of the passive grammar rule is to drop a trace into the input buffer; the trace acts as a dummy NP, a placeholder for the missing NP Object of *kiss.* The pattern for such an action is roughly, an active node of VP, labelled "+ed" and an some record that passive *be* has been encountered. How is this done? By assumption, the VP node is labelled with the names of rules that attach items to it. In this case, for example, the VP will be labelled with the name of the rule that attached the verb *kissed* to it. Similarly, the S node (actually, the maximal projection of Inflection) will

32. For the annotation approach and the second justification I am indebted to M. Marcus.

be labelled with the names of the rules that attached the NP Subject and some form of the verb *be*[33]

As a result, the passive rule will require that the left-context of the parser contain an S node marked *+be* and a VP node marked *+ed*. Why is this important? Later examples, such as, *Sally was kissed by Bill* force a rule generalization so that passive triggers on a buffer pattern of the following form:



That is, the rule will trigger on any kind of element in the buffer. Given just the buffer pattern then, the passive rule would trigger -- erroneously -- in a sentence such as, *Sally has kissed John*. What prevents it from doing so is the left-context of the parse, stored as part of the rule trigger. In the case at hand, the cyclic node above the VP will not be annotated with the feature *+be*, thus blocking rule application.

As an example of the second function of annotation, consider the yes-no question, *Will Bill kiss Sally?*. According to the actions of the grammar rules acquired by the system (see later in this chapter), a *switch* is used to convert this sentence into the form, *Bill will kiss Sally*. Then the parse proceeds as in a simple declarative sentence. But if this is so, then there would be no way to distinguish between the structure built by this sentence and its declarative counterpart. Therefore, any later or concurrently active semantic interpretation routines would be unable to distinguish between declarative and question forms. Since these sentences obviously differ in meaning, there must be some way that the parser marks them as different. A straightforward marking method is the annotation device. The yes-no question will mark the S node with the name of the switch rule, while the declarative sentence will not. (Note that the currently active node at the point where the switch is performed is in fact S.)

4. Specific and General Rules

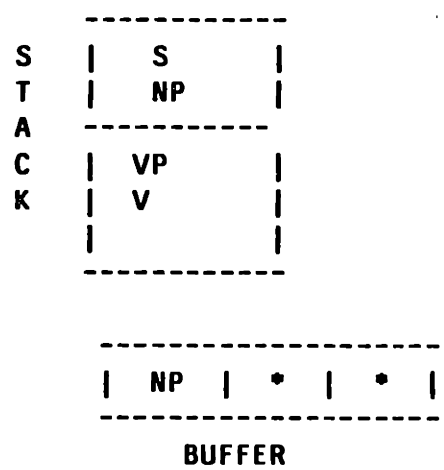
The passive rule illustrates one remaining complexity in the way that grammar rules execute. Suppose that the passive rule has triggered, and has dropped an NP trace into the buffer. Presumably, the next

---

33. This annotation also is automatic. By convention, and in accord with ideas developed by Williams [1981b], the features of the Head of a phrase are percolated through to the projections of that phrase. For instance, the features of INFL, by assumption the Head of the Sentence phrase, are automatically passed through to the S node; the features of a Verb are passed through to label the VP node. For further details, the reader is referred to the next section on the role of base phrase structure rules.

This percolation mechanism has also been adopted in recent theories of word formation; see Lieber [1980]; Farmer [1980].

rule that should execute is one that will attach the NP trace to the VP node, as the argument of the V. What is the pattern of this rule? It is just the ordinary NP Object attach grammar rule, and so must have a pattern something like this:



But then, we now have two rules that match after the NP trace is dropped: this Object attachment rule and the passive rule. To block this possibility, the following criterion is invoked: *Specific rules execute before general rules*, where "more specific" means "has more specific predicates specified by its buffer pattern." In the case at hand the Object attach rule is "more specific," since it calls for the element filling the first buffer position to be at least an NP, whereas the generalized passive rule demands nothing at all about the first, or any buffer position. Therefore, the Object attachment rule will execute, attaching the NP to the current active node, the VP. As usual, the system will then check to see whether the current active node has been completely built; in this case it has, since the subcategorization frame of the Verb is satisfied and the input buffer holds the end-of-sentence marker in its first slot. As a result, the completed node is dropped into the input buffer, deactivating the rule packet that holds the passive rule; the passive rule will never get a chance to repeat its execution in this phrasal domain.

There are other situations where this ordering principle applies; one is covered in Section 6 of this chapter, in a discussion of the acquisition of so-called "Diagnostic" rules. In particular, the proper ordering of the rule to deal with Imperative sentences and the rule to deal with Subject-Auxiliary verb inversion can be established by reference to the specific-before-general protocol.

The requirement that specific rules execute before general rules is a familiar one in the literature on production systems; see McDermott and Forgy [1978] and Rychener and Newell [1977]. It is also a principle that has been recognized in the linguistic literature. For example, Kiparsky [1973] (for

phonological rule systems) and Lasnik and Kupin [1977] (for transformational rules) both propose such a condition.

Perhaps more importantly, the specific-before-general protocol can also be justified on grounds of acquisition, via the Subset Principle introduced earlier. To avoid over-generalization in the face of positive-only evidence, it is crucial that an acquisition procedure guess the narrowest class of languages consistent with the evidence it has seen so far. Translating this desideratum into the domain of grammar rules, it would appear that in general a buffer pattern that is more specific has a smaller range of application, hence allows a smaller set of surface strings. If this rule is to be followed, then it is apparent that a specific rule should be allowed to fire before a more general one; otherwise, the general rule will mask the effect of the specific rule.

But there is no facility for even representing extrinsic rule ordering information in the system, as currently designed, let alone learning this information.[34]

As Baker [1979] observes, any such extrinsic ordering may demand negative evidence for its acquisition. In a system where rules can be optionally or obligatorily applied or ordered, then in order to learn that Rule A must precede Rule B an example must be presented showing the mistake of the reverse B-A ordering. But negative examples have not been permitted; thus, there must be some other means whereby it just so happens that A fires before B. The specific-before-general scheme is one such mechanism for ensuring that rules execute properly.[35] In short then, the rule ordering principle is justified both from the standpoint of acquisition and from an engineering point of view.

---

34. Note that there is no rule priority system explicitly encoded into rules, as in the original Marcus design, nor any rules that say that a certain other grammar rule *must* be executed next.

35. Indeed, without such a principle it is difficult to see how one of Marcus' diagnostic rules could ever trigger without some explicit information about rule ordering.

To elaborate, a diagnostic rule is one that decides among one of several (usually two) alternative parsing actions. For example, the rule that decides whether *have* is a main verb or an auxiliary verb in, *Have the boys...* by looking at the morphology of the verb following the NP is a diagnostic rule.) As mentioned, the acquisition of such a rule is discussed in Section 6 of this chapter.

2.4 The Parsifal Interpreter and Phrase Structure Acquisition

So far the three components of the acquisition model have been outlined -- the initial state of the system's knowledge, the input data it exploits to acquire new knowledge, and its acquisition procedure. Two sorts of knowledge about syntax are to be acquired, corresponding roughly to the base and transformational components of a generative grammar. The current section covers how the model handles the acquisition of just the first type of syntactic knowledge.

The first job of this section is to show how the factoring of syntactic knowledge implicit in a division into base and transformational components can be grounded in a revised Parsifal parser. This is not simply an academic exercise. The purpose of associating distinct functional tasks of a generative grammar with distinct sub-parts of Parsifal is to allow one to focus on the acquisition of the sub-parts one at a time. The goal will be a system where, for example, individual lexical entries can be identified with Parsifal actions and data structures distinct from those Parsifal objects associated with more "transformational-like" operations. Note that this modularization is not a "clean" one: the same basic Parsifal operator/data structure (such as a grammar rule) may sometimes serve to implement lexical knowledge and at other times may mirror transformational effects. However, the actions of the grammar rules in the two cases will turn out to be distinct, and so one will be able to study the acquisition of lexical insertion and transformational-like rules separately.

The factoring strategy adopted here roughly parallels that outlined in Chomsky's *Aspects of a Theory of Syntax* [1965], Chapter 2. An *Aspects*-style generative grammar partitions its base component into two sub-parts: (1) context-free phrase structure rules, specifying the permissible hierarchical arrangements of individual terminal items (essentially words and grammatical morphemes) into phrases and (2) lexical insertion rules, specifying the contexts under which actual terminal items (words and morphemes) may be inserted into a phrase structure tree. It demonstrates how a combination of the single Parsifal grammar rule action *attach* and the Parsifal packet system serve as the functional analogue of the lexical insertion portion of an *Aspects*-style base component. By factoring out lexical insertion contexts in this way, one can begin to formulate a theory for the acquisition of lexical knowledge with respect to syntax. In the syntactically-oriented picture, the construction of new lexical entries ("learning new words") reduces to acquiring at least the proper conditions for lexical insertion, including the features exploited for context checking. To set the stage for this discussion, this section continues with an analysis of the structure of lexical entries in the Parsifal model, their interaction with phrase structure rules, and how new or modified lexical entries can be acquired during the on-line, left-to-right parsing of positive example sentences. The section concludes with a case study of lexical acquisition, examining the acquisition of the English Auxiliary verb system. A corresponding formal analysis of this procedure is provided in Chapter 3.

If *words* were the only elements of syntactic knowledge, then a theory of lexical acquisition might be

sufficient to account for the acquisition of that knowledge. But words are also hierarchically (and linearly) arranged by grammars into *phrases* -- the information associated with the context-free phrase structure rules of the base component of a generative grammar. The acquisition procedure must learn what system of hierarchically arranged structures is imposed on top of its basic knowledge of words. In large part, the model of the acquisition of this knowledge is based upon the so-called X-bar theory of Chomsky [1970] and Jackendoff [1977]. The X-bar theory has been standardly used as a means for describing regularities among base phrase structure rules; it shows how all such rules can be expressed as the instantiation of just a few basic skeleton patterns, a crude operator-operand structure. By reducing the description of a possible phrase structure rule to the combination of a (fixed) universal template structure plus the setting of a small number of "open parameters" that determine the order and hierarchy of phrasal combinations, the task of acquisition is enormously simplified. The heart of the phrase structure acquisition component is therefore a description just what the universal templates and open parameters of phrase structure rules are.

This theoretical discussion is followed with a discussion of the implementation details describing the way in which these X-bar "templates" are encoded in the Lparsifal packet structure (essentially, a parameterization of the packet system of Parsifal). The chapter concludes with a sequence of example scenarios that demonstrate the capabilities of the phrase structure acquisition procedure and the value of the X-bar theory.

To summarize, the goal of this section is to argue for a division of Parsifal's computational abilities into three sub-parts, each a distinctive combination of grammar rules and the packet system:

> (1) Lexical insertion.
> This component consists of the grammar rule action *attach*, plus local context-checking implemented by grammar rule patterns and the packet system. An Example:
>
> IF [first item in the buffer is a Noun]
> AND [current active node is a Noun Phrase]
>
> THEN [attach first item to current active node as a Noun]
>
> ONLY WHEN
> [active packet is Parse-Head]

(2) Base phrase structure order.
Parameterized packet template system, in conjunction with the grammar rule action *attach*. Example (the rule S-->NP VP)

IF [first item in buffer is NP]
[second item in buffer is V]
AND [current active node is S]

THEN [attach first item to current active node as NP]

ONLY WHEN
[active packet is Parse-Specifier of S]

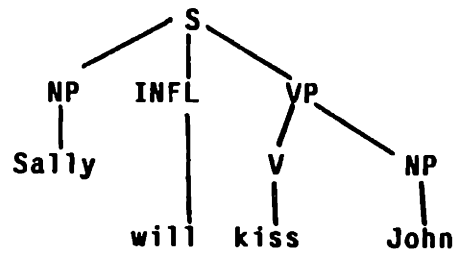
(3) Transformational-like operations.
Inversions, movements, and insertions via grammar rule actions *switch* and *insert trace*, and *insert lexical item*. Example:

IF [first item in buffer is an aux verb]
AND [second item is an NP]

THEN [switch first and second items]

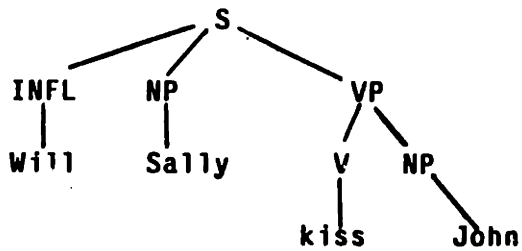
2.4.1 Parsifal and Base Phrase Structure Rules

Our first task is to model some of the division of labor among the components of a generative grammar with separate data structures in Parsifal, decomposing Parsifal one step beyond a two-way split into interpreter-plus-grammar rules. To begin, recall that a generative grammar is conventionally characterized as an ordered sequence of distinct linguistic levels, with mappings describing the relationship between levels. Central here are two representational levels, *base structures* (or the level of "phrase markers") and *annotated surface structures*. The base component of the grammar delimits the set of possible base structures. As described in Chomsky's *Aspects of the Theory of Syntax*, it consists of two sub-parts: (1) a set of context-free phrase structure rules and (2) a set of lexical insertion conditions. The transformational component is a function (generally decomposed as a derivational sequence of more elementary functions) that takes the structures so generated into annotated surface structures. To take a standard example, the base component might generate a tree like the following,



(a) Schematic of tree generated by base component.

which could be mapped to:



(b) After transformational mapping.

On this analysis one can see that the base component performs several distinct tasks. The context-free phrase structure rules delimit the possible hierarchical and linear arrangements of syntactic phrases available as input to the transformational component -- essentially the branching structure of trees. In addition, the base describes the labeling of phrasal nodes and the conditions for insertion of lexical items as the terminals of trees. These tasks of phrasal structure and labeling have conventionally been assigned to a set of context-free re-writing rules that dictate the order and branching of syntactic categories. For example, the rules S⇒NP INFL VP; NP⇒Noun would begin to describe the tree of Part (a) above.

The methods for lexical insertion contexts have typically been classified as either strict subcategorization or selectional restrictions (Chomsky [1965 page 95]). The distinction between these two is intended to capture two apparently different kinds of co-occurrence relationships that may exist between the words of a sentence, as indicated by the following examples:

(i) Strict Subcategorization:
John kissed the girl with blond hair.
*John kissed.

(ii) Selection:
John kissed the girl.
*John kissed the idea.

The second sentence of the first pair above is presumably ill-formed because *kiss* is a transitive verb and must appear with a Noun Phrase Object. This sort of restriction, based on the presence or absence of certain nodes of a phrase structure tree (in this case, a Noun Phrase node), is a strict subcategorization condition. But the verb *kiss* requires in addition that its Noun Phrase Object be in some way marked as "Animate" (unless taken in a metaphoric sense); one cannot kiss an idea. This restriction, based on the syntactic features of a category node rather than simple existence or non-existence of a node, is selectional.

In Chapter 2 of *Aspects of the Theory of Syntax* [1965], Chomsky proposed a way to factor these subcategorization conditions out of the purely phrase structure rules of the base, modularizing the base along the functional lines outlined above. One sub-system, the categorial component, is left with the task of determining just the branching structure and linear arrangements of phrasal categories in a language -- for example, the fact that English sentences consist of Noun Phrase Subjects followed by a Verb Phrase or a Predicate Phrase. The remaining sub-system, the lexicon, incorporates subcategorization conditions, and has the job of determining when a lexical item may be inserted into a given phrase structure tree. The modularization is important because it provides a further way of dividing the grammar rules of Parsifal into functional classes. Some grammar rules are used to determine strictly the branching and order of categories (for example, that the parsing of a Verb Phrase must follow that of parsing a Subject Noun Phrase), whereas others serve to determine lexical insertion (for example, that an Animate Noun can be part of the Object Noun Phrase of the verb *frighten*).

Chomsky's partitioning scheme is as follows. To reproduce the effect of context-checking, one introduces context-sensitive rewriting rules that encode additional *complex symbols* onto the "standard" phrasal categories like "N" or "V" so that contextual environments can be accessed as a derivation proceeds. In essence, the appropriate strict subcategorization or selectional context is stored with each lexical entry, and then lexical insertion is permitted only in contexts where the features on a lexical entry match the context conditions of the rewrite rule. To take a simple example, consider again the strict subcategorization condition that transitive verbs such as *kiss* and *hit* require a Noun Phrase Object, whereas other verbs, such as *die*, do not. Transitive verbs can then be stored in the lexicon perhaps as, {*kiss*, [+V], context: ___NP,...}, and intransitive verbs as, {*die*, [+V], context: ___#}.

Call these entries *complex symbols*; the context sub-entries are often referred to as *subcategorization frames.*[1] Now the rewrite rule,

VerbA⇒Complex Symbol/___{NP or null}

where A⇒ B/X___Y means "re-write A as B in the environment X___Y"

expresses the desired subcategorization condition (given the convention that insertion can take place only if the given "context" features of the lexical entry match those of the rewrite context).[2] Further, since the Verb must have been introduced by some rule of the form Verb Phrase⇒...V..., we can add the following condition:[3]

VerbA⇒Complex Symbol/___{NP or null}, where V{NP or null} is a VP.

Finally, it is apparent (as Chomsky also shows) that the method of complex symbol encoding can be extended to exploit syntactic features such as *Animate*, and so incorporate selectional restrictions as well; the details will not be covered here.

One property of the complex symbol method that will prove important for Parsifal is that the "rule" of lexical insertion assumed by that approach has a kind of transformational power. This is because the insertion rule (encoded either in some combination of rewrite rules, lexical entries and labelling conventions, or lexical entries alone) must be able to examine portions of tree structure (phrase markers) in order to do its job. For example, the subcategorization frame for Verbs must look to see whether a Noun Phrase node lies to its immediate right, exactly as, say, a Subject-Auxiliary Verb inversion transformation might.[4]

---

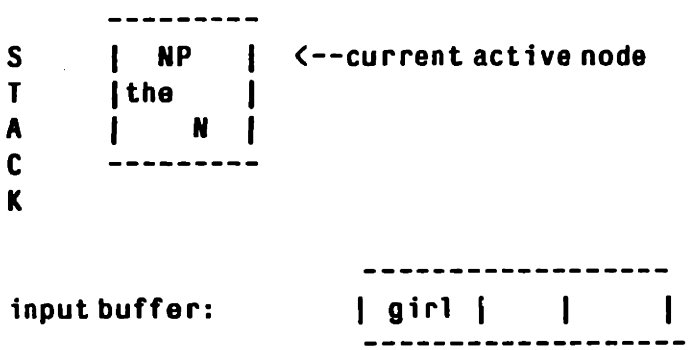
1. Here, the notation "___" indicates the locus of insertion, while "#" represents the null string).
2. Actually, since the context itself is stored with the lexical entry, it need not be restated in the rewrite rule itself; one can supply the context condition by convention and give the rule simply as, "V⇒Complex Symbol". (Chomsky [1965 page 100]).
3. Formally, Chomsky states this additional condition on subcategorization as:
A⇒complex symbol/X___Y where XAY is a Z
and where Z is a category symbol that appears on the left in the rule Z⇒ ... A... that introduces A.
[1965 page 99]
The significance of the added "is a" condition will become clear when the relationship between lexical insertion and Parsifal grammar rules is established.
4. Chomsky observes that the lexical insertion transformations are also strictly local in that they rewrite a single left-hand category symbol -- "V" in the case above -- on the basis of a tree structure context that includes the node being rewritten. (1965, Chapter 2, footnote 18). Joshi and Levy [1977b] have formalized this notion of local transformation, extending a result of Peters and Ritchie [1969] that shows that such rules when used for *parsing* can (weakly) generate only context-free languages. This result is of some computational significance in that, as Joshi and Levy show, one can then rely on any standard context-free parsing algorithm to "implement" lexical insertion transformations. This mathematical result is a key to some of Parsifal's success, although Marcus was apparently unaware of the Peters and Ritchie work at the time Parsifal was written.

Summarizing the discussion so far, one can usefully isolate two sub-systems in the base component, one fixing hierarchical and ordering structure (context-free phrase structure rules) and one the conditions for lexical insertion:

> Along with a lexicon, then, the base component of the grammar contains: (i) rewriting rules that typically involve branching and that utilize only categorial [...] symbols and (ii) rule schemata that involve only lexical categories, except in the statement of context... The rules (i) are ordinary phrase structure rules, but the rules (ii) are transformational rules of an elementary sort.
> [1965 page 98]

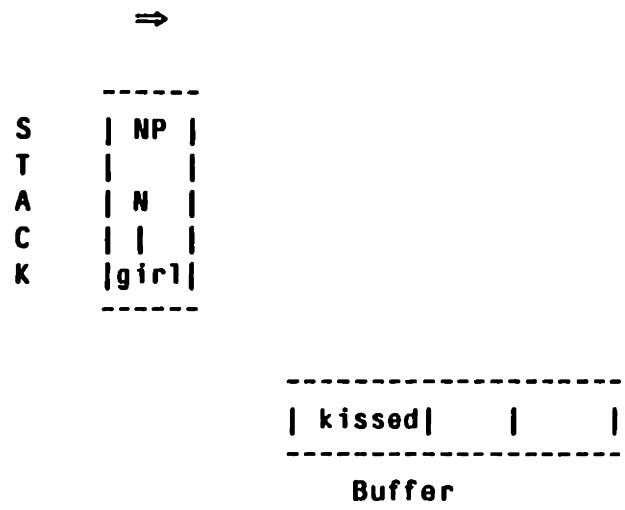
In keeping with the goals of this section, we must now show that a distinctive combination of Parsifal operations and data structures can be identified as the functional analogue of lexical insertion contexts.

The basic Parsifal command that dictates the insertion of lexical items (and phrases) is the grammar rule action *Attach*. *Attach* moves a specified item in some input buffer cell and appends it as the daughter of an accessible node in the active node stack, invariably, the current active node ("top-of-stack"). For example, suppose that the following tree, a fragment of a Noun Phrase, had already been built, and that the input buffer held the token "girl": in its first cell:



Further suppose that the lexical entry for "girl" states that it is a Noun (marked, say, +N). The following grammar rule *attach* operation would move *girl* from the buffer to its proper position underneath the emerging Noun Phrase tree:

Attach first item in the buffer to the current active node as a Noun.



This Parsifal grammar rule action clearly has the same *effect* as lexical insertion: a single node (a Noun) is re-written as some specific lexical item. In addition, as with all Parsifal grammar rules the *attach* action may be conditioned on the basis of a pattern, a predicate true of the features of items currently in the buffer, the current active node, and the current cyclic node. In the case of the Noun Phrase construction above, one obviously wants to attach *girl* to the current active node only if (1) one is rewriting a Noun Phrase as a Noun (the current active node has the feature "NP"); (2) the first item in the buffer has the feature *Noun*; and (3) the NP has already been expanded so that the proper insertion context for a Noun -- a Determiner for common Nouns, none for Proper Nouns -- lies to the left. Thus the full Parsifal grammar rule for this attachment should read something like this:

```
IF      [current active node is an NP]
AND     [first item in buffer is a Noun]
AND     [Determiner has been attached as a daughter of the NP if
         item in the buffer is a common Noun;
         else nothing has been so attached]

THEN    [attach first item to current active node as a Noun]
```

This is clearly just an alternative formulation of the context-sensitive re-writing rule:

N⇒Complex Symbol/A___; where A is an NP.

sample lexical entry: {girl [+Noun], context:[Determiner___],...}

```
{Sue [+Noun], context:[#___#],...}
```

However, Marcus actually wrote a pair of grammar rules to handle this case:[5]

```
RULE NOUN IN packet PARSE-NOUN

IF      [first item in the buffer is a Noun]
THEN    [Attach first item to current active node as a Noun]

RULE BUILD-NAME in packet PROPER-NAME

IF      [first item in the buffer is a Proper Noun]
THEN    [attach first item to current active node as a Name]
```

In these two grammar rules there is apparently no explicit check that the current active node is a Noun Phrase, or whether there is a required Determiner to the left (if the item to be attached is a Common Noun), or nothing at all (if it is not).[6] Rather, this information has been encoded via the packet associated with each grammar rule and the possible pathways that can lead to the activation of a packet. Recall that a grammar rule can trigger only if its associated packet is *active*. Crucially, the packeting system can be arranged so that the packet *Parse-Noun* is active just in case a Noun Phrase node has been created and a Determiner has been already attached to it (again ignoring the complication of quantifier phrases), while *Build-name* is active if there is no Determiner and the Noun is marked -Common. A more complete version of the packet network as actually implemented in Marcus [1980] (incorporating possible quantifier and adjective paths before a Noun but omitting some of the complications of Noun Phrase specifiers like possessives) looks roughly like this:

---

5. Marcus' grammar rules actually do something else: they create a new phrase structure node, N. This suggests that the labeling function of phrase structure rules might itself be accomodated in the lexicon. This suggestion is taken up in detail below.

6. Neglecting for the moment the complications of adjective quantifier phrases, as in *too many girls*...

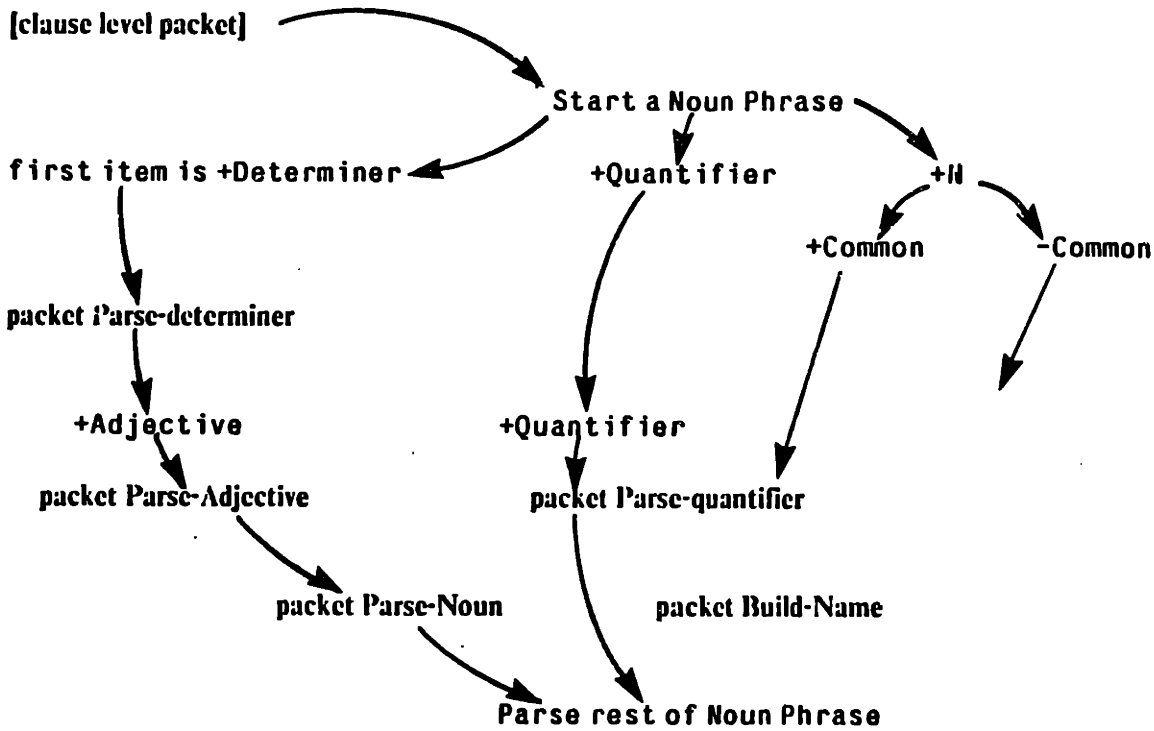


Figure 2.1 - Schematic of Parsifal packet network for a portion of the Noun Phrase system.

The only way that Marcus' system (for English) can ever arrive at a state where the packet Parse-Noun is active is for the context-check for an NP and a Determiner to have been automatically satisfied. As a result, there is no need to perform this now redundant test in the body of the grammar rule itself.[7]

It is clear then that the grammar rule *attach* action, grammar rule patterns, and the packet system act in concert to reproduce the effect of the context-sensitive lexical insertion rules as described in Chomsky [1965] -- at least in the case where actual lexical items must be attached to the annotated surface structure tree. The explicit correspondence is as follows:

7. As noted by Swartout [1978], a single packeting network working in conjunction with transition tests based on the current state of the parser (like the one sketched above) is formally a (Recursive) Transition Network (RTN).

```
Aspects:   Action: A⇒"complex symbol"
           Context Conditions: /X___Y; XAY is a Z

Parsifal:  Action: Attach "complex symbol" as a node of type A
           Context Conditions: X=complex features of
           current active node
                              or cyclic node;
                           Y=complex features of buffer items;
                           Z=rule packet name
                           (proper path of activation)
```

In effect, the packet name stores some of the information about the derivation up to that point. For instance, the activation of the packet Parse-subject actually means that the parser is at the point in the derivation where the subject NP is being analyzed. [8]

The detailed insertion context information stored with lexical entries permits us to eliminate rewriting rules of the form, "N⇒...some lexical Noun", since these are now completely redundant. Alternatively, one can store these patterns as node admissability conditions; this is closer to the actual form of grammar rules.[9]

If context frames can be used in this way to replace phrase structure expansions of the category "N", one might wonder whether higher categories such as Noun Phrases are also subject to the same replacement technique. This eventuality too was anticipated in Chomsky's *Aspects*. Continuing from the last cited quotation, Chomsky added the following comment:

> One might, in fact, suggest that even the rules (i) [ordinary phrase structure rules] must be replaced, in part by rule schemata...or local transformations.
> [1965 pages 98-99]

As an example, consider the environment for the insertion of Noun Phrases into the Subject position in English sentences. The conditions are roughly as diagrammed below, where "X" represents the locus of insertion:

---

8. Compare this to the more familiar claim about an LR(k) parser, namely, that the top-of-stack symbol encodes all the information about the parse required to uniquely determine the next move of the parser. (This is because the sequence of sentential forms in a right-most derivation of any context-free grammar can be generated by a right-linear hence, regular grammar; as is well known (De Remer [1969]), this fact allows us to encode in the top-most stack symbol the state the parser would have been in had it scanned the entire stack. Thus, even though the left-most context is potentially infinite, what is relevant for parser control can be finitely represented. Similarly, even though the active node stack of Parsifal can grow arbitrarily, the patterns necessary to determine the parser's next move are finite. See Chapter 5 for additional discussion.
9. For discussion, see Chomsky [1965 fn. 18 to Chapter 2].



In other words, one situation in English a Noun Phrase can be inserted is when (1) the local context is [__{+V}], where {+V} can be any complex symbol having the feature +V(erb), i.e., a Verb or a Verb Phrase; and (2) the NP is immediately dominated by an "S" node. Thus, [__{*is:* +V}] or [__{VP: kissed Sue}] would both be appropriate local contexts for the insertion of a Noun Phrase, if the NP-V(or V plus additional material) are also known to form an "S"-type node (that is, if something like the rule, S⇒NP VP is assumed). Referring to Chomsky's formulation of a context sensitive rewriting rule above, one can see that this condition for Noun Phrase insertion is an extension of the lexical insertion rule schema to a situation where the definition of "complex symbol" has been expanded to include phrase structure category labels like NP or VP.[10]

Having shown that the task of lexical insertion is carried out in the Parsifal by the analogue of context-sensitive re-writing rules, it is perhaps not surprising then to discover that the same combination of *attach* actions plus packet activations can be used to duplicate the effect of phrase structure construction -- given the re-interpretation of "complex symbol" discussed in the previous paragraph. For example, here is the Parsifal grammar rule that attaches a completed Noun Phrase constituent (in the first buffer cell) to the current active node:

```
{RULE UNMARKED-ORDER IN PARSE-SUBJECT
IF      [item in first buffer cell is a Noun Phrase]
AND     [item in second buffer cell is a Verb]

THEN    [attach item in first buffer cell to current
      active node as a Noun Phrase]}
```

Just as in the case of lexical items, one can view this grammar rule as a context-sensitive admissability condition on the valid insertion of Noun Phrases into the currently being constructed parse tree. Similarly, the import of the packet condition here is again simply to record the fact that an S (Sentence) node is the current active node, and hence must immediately dominate any node attached by the above grammar rule.

---

10. Actually, under the X-bar theory of phrase structure described in the next section, it will be shown that the labels such as "NP" or "VP" are literally "complex symbols" consisting of bundles of features just as "boy" or "kiss" are.

Parsifal has one other way of encoding phrase structure rules besides the *attach* action working directly with the packeting system. Phrasal nodes can also be created immediately upon exiting or entering a packet of a given type if the phrase structure of the grammar demands that this be so. For example, take the Unmarked-Order grammar rule above. The full statement of this grammar rule contains two operations executed in addition to the attachment listed above, as indicated in boldface:

```
{RULE UNMARKED-ORDER IN PARSE-SUBJECT
IF      [item in first buffer cell is a Noun Phrase]
AND     [item in second buffer cell is a Verb]

THEN    [attach item in first buffer cell to current active node
      as a Noun Phrase]
        [Deactivate packet Parse-subject. Activate packet
      Parse-Inflection]}
```

That is, rules in packet *Parse-Inflection* will now be eligible to execute. But Marcus has arranged it so that, of all the grammar rules in the Parse-auxiliary-verb packet, one must fire first: a grammar rule that creates a new Inflection Node. In short, once the S node is partially expanded as "NP...", it must be expanded as "NP INFL..." as well. (However, in Marcus' published grammar, the Inflection node may need not dominate any lexical material.) These sorts of grammar rule actions do not attach any items from the input stream to the parse tree. Rather, they determine only the possible order of phrasal categories, for example that a Verb Phrase can be expanded as Noun Phrase-Verb as opposed to Verb-Noun Phrase.

It should now be apparent that modularizing syntactic knowledge in this way leaves very little work for the phrase structure rules to do (compared to the formulation in *Aspects*). All information about what categorial types are allowed in a certain phrasal context, whether phrases are optional, and so forth may be listed in lexical entries. One sort of information remains: the lexicon does not specify what might be called the "pure" phrasal structure of base phrase markers as exemplified by the last grammar rule action discussed above -- the linear order of phrases and their internal branching structure, roughly, the skeleton shapes of possible phrase structure trees.[11] To take another example of this type, consider a verb such as, say, *give*, and suppose that its subcategorization frame is something like, [___NP, ___*to* NP] (that is, assume that part of the "meaning" of *give* is at least *give*: [something to someone]. The exact details of the frame do not matter here.) Note that while this entry does say that the Verb *give* must be followed by an NP and *to*-NP phrases, it does not specify the order of these

---

11. See Hale [1979]; Heny [1979] for original proposals to this effect.

two phrases with respect to one another. That is, the lexical entry as stated for *give* permits either the expansion VP⇒V NP *to*NP or VP⇒V *to*NP NP. Other principles of grammar -- in particular, an adjacency constraint on assignment of case to NP arguments -- determines phrasal order in this case. The theory for the acquisition of this purely "ordering" knowledge is based upon the X-bar theory of phrase structure, and is covered below.

To summarize, the Parsifal grammar rule *attach* and the packet system comprise a computational analogue of the base component of a transformational grammar. There are essentially three different ways that the grammar rule and packet system are combined to reproduce the effects of a base component: (1) the effect of context-sensitive lexical insertion is obtained via a combination of the *attach* operation and the packeting system. (2&3) the effect of phrase structure expansions is obtained via context-sensitive tree admissability conditions based upon either (i) an *attach* operation, using local contextual information and the encoding embodied in the packet system, or (ii) transitions between packet types. The knowledge embodied in grammar-rule/packet combinations (1) and (2) directly exploits the syntactic features of individual lexical items (and by inheritance, phrases), and so may be stored directly with each lexical item. The acquisition of this knowledge is modelled in Lparsifal by exposure to positive examples, with subsequent collapsing of redundancies via feature generalization; this acquisition will be discussed immediately below. The knowledge in rule-packet combination type (3) makes use of a set of parameterized templates for phrase structure, the parameters being roughly the order and branching structure of categories like NP or VP. Here acquisition is largely reduced to setting the "parameters" (as yet unspecified) that determine the skeleton shapes of phrase structure trees.

Finally, it is important to stress that the *attach*-packet system combination accounts for the bulk of grammar rules in Marcus' grammar. Of the hundred or so grammar rules in Marcus' published grammar [1980 Appendix D][12] roughly one half fall under categories (1)-(3) above. That is, a majority of the grammar rules written by Marcus can be interpreted as immediately re-writing a single node in a specified context (they are "local transformations"). The remaining grammar rules of Parsifal can be broadly classified as one of four types:

(1) Inversions (e.g., Subject-Auxiliary inversion)

(2) Insertions of specified lexical material. (e.g., *You*)

---

12. This count excludes special-purpose rules written by Marcus to handle peculiarities of time expressions and dates, case-frame interpretation rules, and so forth.

(3) Movement of Noun Phrases and *wh* constituents, e.g., fronting of Noun Phrases, (including passive) and *wh*-questions.

(4) "Diagnostic" rules, specialized rules that, for the most part, choose between alternative categorizations for the same lexical item; e.g., *Have* can be either a Main Verb or an Auxiliary Verb in *Have the boys....*

The current section is concerned with the acquisition of the roughly half of the Parsifal grammar rules that involve context-sensitive lexical insertion and phrase structure construction. Acquisition of the remaining types of grammar rules is the topic of Section 6.

2.4.2 The lexicon and lexical acquisition

Though it is easy to demonstrate how rewriting rules may be replaced by context conditions stored with lexical entries, that does not eliminate the problem of acquiring that knowledge, whether as rewrite rules or lexical structure. Glancing at the form of the lexical insertion conditions, one can see that their acquisition amounts to the acquisition of three pieces of knowledge: (1) the proper features of items to the right and (2) the left of the insertion, and (3) the proper packet (as a proxy for the proper sequence of active packets leading to the insertion). The triggering experience for the construction of an entry is presumed to be exposure to positive examples. Let us see how this might work in a simple case, and then backtrack to explore the problems that arise and how they might be handled.

First of all, one is faced with the problem of how the acquisition procedure "gets off the ground." How does the acquisition procedure know that, say, *kiss* is a Noun in a sentence such as, *I gave Bill a kiss*, but a verb in the sentence, *I kissed Bill*? An often-made assumption (see, e.g., Limber [1973]) is that one can assume that Nouns denote objects in the physical world, and verbs, actions.[13]

At first glance, this approach still seems problematic. The difficulty is that many tokens of English are homophonically both nouns and verbs, e.g., *kiss Sally, a kiss.* How is the acquisition procedure to know the difference between these? Again, one can resort to a principle of inference from clear cases. While it is true that many tokens have homophonic variants, many do not. For example, names

---

13. Presumably, there is some sort of "cognitive prerequisite" for understanding that a linguistic token can denote an object in the world, and this fact that led some psychologists to conflate the acquisition of linguistic knowledge with that of other conceptual systems – e.g., Piagetian notions of "object permanence." It is easy to see how this confusion could arise. If it is true that the notion of physical object plays a role in grounding certain lexical categories, then one might expect to find a correlation between, on the one hand, the developmental of object permanence; and, on the other hand, the use of, say, single linguistic tokens as names. But there is, to my knowledge, any explanation as to how this connection comes about. The linguistic system bottoms out in mystery.

cannot be verbs; *John* and *Sally* can never denote actions. And this is not a rare phenomenon. Other common nouns, such as *daddy, cat, boy, girl,* always appear as nouns. Similarly, there are tokens that are always verbs, at least in input that will be plausibly be available to a child: *eat, give,* and the like. All that is needed for the system to proceed is that there be one clear case that establishes the proper argument structure for verbs in English, e.g., a sentence such as, *Mary ate candy.* As will be discussed in more detail below, this fixes the canonical complement structure for Verb Phrases for English, in the pattern V--NP. What happens if a sentence is encountered where the categorization of an item is uncertain, as in, *Bill kissed Mary*? (We put aside for the moment the tense feature on *kiss.*) *Kiss* could be projected as either a Verb Phrase or a Noun Phrase. If basic sentence structure rules have not been established, so that the English constituent order NP--INFL--VP is unknown, then these two possibilities cannot be directly ruled out; thus, we have a case of unclear inference, and the acquisition procedure will simply avoid drawing any conclusions from this case.[14] Once basic constituent order has been established by a clear case, then repetitions of previously "too complex" examples may now lead to success. (Note that the system does *not* store the example sentences it receives, and so does not have memory for past data.) Several cases of this "intrinsic" ordering effect are provided in Sections 5 and 6 of this chapter. To take a simple example, suppose that basic constituent order is established. Now suppose that the system receives the example *Bill kissed Mary* again. This time, the assignment of the feature +V to *kiss* meets the constituent structure constraints already acquired, so the acquisition procedure can incrementally construct a new grammar rule to deal with *kiss.* Analyzing *kiss* as a Noun still requires more than one new rule, and so is avoided. Similarly, consider how the system might learn that *kiss* can be a Noun in the string, *I gave Bill a kiss.* Plainly, the system can first learn that *a* unambiguously announces the start of a Noun Phrase, via such examples as, *I ate a candy.* This, plus the knowledge that *give* has a subcategorization frame roughly like, give:[NP, to NP], is all that is needed. Then, *kiss* could be either +N or +V, but only the +N feature is compatible with required context.[15]

---

14. This will be demonstrated in detail below.

Note that it might be possible to make a decision even in this case, if the sentence is associated with a thematic representation assumed known from extra-linguistic context. But we will put aside this possibility here to pursue another line of argument.

Also recall that just because the acquisition procedure does not provide a full syntactic analysis of a sentence does not mean that the sentence remains "uninterpretable." It may very well be interpretable by other means -- e.g., the predicate-argument structure could be reconstructed.

15. Put another way, co-occurrence restrictions are expressed via node admissability conditions, in the sense of Peters and Ritchie [1969], rather than generatively. The acquisition system fixes the admissability conditions for the insertion of lexical items. Just those feature bundles that meet admissability conditions "survive" the feature checks required by local context. One advantage of this approach is that it is neutral about the status of phrase structure rules and non-terminal labels. On this account, there is no distinct set of context-free re-write rules that form a grammar for generating a language. Rather, there are conditions on whether items meet their local context checks or not. If little or nothing is known about the features of a lexical item, this process can actually be used to force the category identity of that item "from the top." Several examples later in Sections 5 and 6 of this chapter illustrate this.

There are processes that can be used to alter categorial identity -- one can, for example, simply add an "ed" ending to verbalize any noun. But this ability is acquired later, *after* basic constituent structure has been fixed, the the significance of the morphological endings established by clearer examples. In short, the acquisition of lexical category information -- when *kiss* is a Noun or a Verb -- seems easier than having to determine categorial identity in the "mature" state, because one is permitted to ignore ambiguous examples so that there are fewer possibilities to consider at any one step. An example of this process is provided later on in this section.

In more detail, lexical entries are formed by two processes of category merger and category splitting, as discussed earlier under the operation of the acquisition procedure's rule generalization protocol. A lexical equivalence class contains all items that "behave alike" under the invariance relation established by the Lparsifal analysis. That is, all tokens that enter in the same way into the construction of parse trees are considered behaviorially identical with respect to the operation of the parser, and hence are placed in the same equivalence class. Note that this condition goes beyond the usual notion of string invariance used to form the equivalence classes of a finite state automaton, in that the operation of the parser is not just one of simply concatenating strings together.[16] Rather, the parser is constructing equivalence classes according to common left-hand and right-hand patterns found in the parser as it analyzes strings, and these patterns can correspond to trees rather than just strings.

An equivalence class of lexical items is formed whenever one has two attachment rules with same left-hand (active node stack and packet) and right-hand (2nd and 3rd buffer cells) context. For details, see earlier in this Chapter, and Chapter 3. So for example, *will, did, could* are all placed in the same class, because they are appear in forms that cause the parser to enter the same state, such as:

I could take the book.
I did take the book.
I will take the book.

The class here is defined by local context, where the notion of "local context" is relative to the Lparsifal parser: namely, the state of the machine at the time the attachment action of *could, will,* or *did.*

A class is split whenever the left-hand or right-hand context of a parse is found to be different for two items in the same class. So for example, the sentences,

---

16. For simple cases, however, such as the INFL system of English, the parser behaves as if it were just concatenating strings. In this case, the invariance relation reduces to that of right-concatenation. This permits one to analyze the operation of the acquisition procedure in this case as an example of finite-state induction. See Chapter 3.

I could have taken the book.
I will have taken the book.

Will cause *could* and *will* to be split off from the item *did*, given that the example *I did have taken the book* is not encountered. Since in fact this ungrammatical sentence will never be encountered, the classes will remain split. Given a covering sample of positive examples, this process of class merger and splitting can be shown to converge to the correct final state (see Chapter 3).[17]

Note that on this view, lexical categorizations are dynamically defined. Indeed, given different presentation orders of examples, one would expect to find stages where categories where incorrectly collapsed together, as in the example above. This would lead to predictable cases of over-generalization in certain early stages, in just those cases where not enough information has been received to distinguish among classes that are distinguishable in the adult state. In fact, this is just the kind of mistake that children often make. For example Baker [1979] observes that children seem to use *it* just like an ordinary lexical NP:

Adult: I turned off the light. I turned the light off.
I turned it off. *I turned off it.

Child: I turned off the light. I turned the light off.
I turned it off. I turned off it.

This effect would be expected in a model where class distinctions are over-generalized by lack of exposure to the relevant cases. Note that this early-overgeneralization does *not* show up as mistakes in grammar rule construction *per se*, but rather in the identification of features that trigger known grammar rules. Crucially, recovery is possible *because class refinement is allowed.* Details of this process are provided below in Section 2.5, where the induction of the INFL (Auxiliary verb) system for English is discussed.

Besides category identity, the "major" lexical items such as Nouns, Verbs, Prepositions, and Adjectives could be considered to possess a characteristic argument structure associated with a kind of compositional semantics. So for example, the verb *want* takes a Noun Phrase Object or an optional sentential complement, as in *I want the book*, or *I want Bill to leave.* How might this knowledge be acquired?

---

17. Thus this approach assumes that at some point all possible distinctions have been revealed by one or another positive example. This assumption is just that of "indirect" negative evidence proposed by Rizzi [1979]: if a construction P has not been found in simple data, assume that P never appears in any data. See Chapter 3 for further discussion of indirect negative evidence.

Again, positive examples can supply most of the evidence about the argument structure of verbs. Suppose, for instance, that an acquisition procedure receives the sentence, *Bill wants a candy.* Then, assuming that a default requirement on adjacency to the verb is being honored -- a default requirement that can be justified -- the trailing NP is assumed to be part of the verb's argument structure. Similarly, examples such as, *I gave a book to Bill* establishes the entry for *give* as, [NP to NP]. The construction of entries is governed by the Subset Principle, described briefly in Chapter 1 and in more detail in Chapter 3. The aim of the Subset Principle is simple. It is designed so that the acquisition procedure will guess as small a language as possible, consistent with the evidence it has seen so far -- so that the procedure will not overgeneralize. This is a necessary and sufficient condition for correct acquisition from positive-only evidence, as discussed in Chapter 3 Chapter 3 also describes the implications of this principle. One implication is that arguments to verbs must be assumed obligatory unless proven otherwise, and that a verb is assumed to have no arguments at all unless proven otherwise. So for instance, *take* will be assumed to have two arguments if an example such as, *I took Bill to the movies* is given first. Later examples, for instance, *I took the book*, easily show that the Prepositional argument is optional. The restriction of argument adjacency is also a default requirement that is invoked by the Subset Principle; if adjacency is assumed to be a requirement for the proper construction of a verb's argument structure then purely structural position (namely, adjacency to the verb) can be used as a cue for argument identification, and the narrowest possible class of output strings is generated. Once this identification has been established, later example (if present) can weaken this assumption, as in, perhaps, Japanese. Note that adjacency is indeed strictly observed in children's simple N-V-N structures.

In Parsifal, argument information was actually encoded in the packet system (as in fact Marcus realized). A different set of grammar rules was activated according to what kind of argument structure a verb could have. Here is a full listing of the Parsifal grammar rule to handle verb complements as published in Marcus [1980]. Packet names are in boldface. (The stylized input language has been slightly modified here in the interests of readability.)

1. If the verb is passive then activate **passive** and run passive rule;
2. If the verb can take an infinitive-object, then:
   2.1 If the verb can take an infinitive without *to*, then
   activate **to-less-infinitive**
   2.2 If the verb can take an infinitive without *to be*, then
   activate **to-be-less-infinitive**
   2.3 If the verb takes 2 objects, one an infinitive (e.g., *persuade*)
   then activate **2-object-infinitive-object**
   else activate **infinitive-complement**;
   2.4 If the verb takes an infinitive object without a subject (e.g., *want*), then activate **subject-less-infinitive-complement**
   else if the verb takes a delta subject (e.g., *seem*)
   then activate **no-subject**.
3. If the verb takes a that-complement, then activate **that-comp**

Rather than storing this information by selecting different packet names -- these are, after all, simply equivalence classes of verb types -- one could choose to store this information with the lexical entries associated with a verb, and then "projecting" this information to the level of structure built by the parser. One can see then, that Marcus has simply built the effects of the Projection Principle into his parser, in that knowledge about subcategorization frames is used to guide the parse.

Later on in Section 6 of this chapter we shall discuss how this system of packet activations can be acquired.

2.4.3 Lparsifal and the X-bar Theory

To accomodate variation in base phrase structure systems -- the order of phrases and their branching structure -- a phrase structure acquisition component must be added to the acquisition model. The obvious candidate for a theory of phrase structure acquisition is the *X-bar* theory of phrase structure rule schemata, originally proposed by Chomsky [1970] and developed by Jackendoff [1977] and others. The X-bar theory proposes that all human phrase structure rules can be derived from a small, finite number of basic tree templates, the *X-bar* schemas. The original motivation for this view arose simply from the study of phrase structure rule systems. Observe, for example, that the context-free rules for Noun and Verb Phrases look very much alike:

Noun Phrase⇒(some leading constituents)...Noun...(PP) (S)
(e.g. *the boy in the park who I know*
Verb Phrase⇒(some leading constituents)...Verb...NP (PP) (S)
(e.g. the boy persuaded John in school that he was ill.

Evidently, both Noun and Verb phrases are constructed around a central "core" scaffolding -- a

single Noun or Verb (roughly, single words), in a uniform kind of operator-operand structure.[18] This simple observation is the essence of the X-bar theory: Chomsky proposed that all phrases (items of a syntactic type) might be formed from simple projections of the categories of certain words (items of a lexical type). What the X-bar theory tells us then is the possible categories of phrases, given a set of possible categories of words (such as Nouns, Verbs, Adjectives, and so forth), and a (simple) set of rules for passing from categories of words to categories of phrases; if words like Nouns and Verbs exist, then so (predictably) may Noun and Verb Phrases.[19] This rule system allows us to collapse the English Noun and Verb Phrase schemas above into the single template below, where "X" stands for a possible lexical category (a Noun or a Verb):

$$X^n \Rightarrow (C_1)...(C_i)X^{n-1}(C_{i+1})...(C_j)...$$
Specifier Head Complement

Let us review the additional X-bar terminology used here. The exponents *n* and *n-1* (n>0) on the "X" category are the so-called bar-levels that indicate the "depth" of a particular tree structure. In practice, *n* is restricted to be a small, positive integer for all phrase structure rules, so that the maximum "depths" of the resulting tree templates are at most a few nodes in extent; an "X" with the maximum index value will be called an "Xmax". (Later on in this section it will be argued that the number of bar levels need not be specified in advance, but in fact can be acquired dynamically, as new equivalence classes of non-terminals are required.) Single lexical items lie at the terminal fringes of the tree, the X^0 level; syntactic phrases are higher-numbered constructs. Consequently, for any given word category of type "X", the X-bar theory makes available phrases of type X^1, X^2,...,X^n.

The indexed syntactic category X^{i-1} on the right-hand side of a rule (dominated by a corresponding X^i on the left) is dubbed the Head of the rule. (The lexical category X^0 is also called the "Lexical Head"; Jackendoff [1977, page 30].) For the purposes of this report, the Head of each rule is presumed unique; that is, there cannot be two or more identically labelled and indexed "X"s on the right-hand side of a phrase structure rule. For example, the following phrase structure rule is invalid:[20]

---

18. One difference between Noun Phrases and Verb Phrases is that NP's do not allow "bare" NP complements at the *surface*, whereas VP's do: *destroy the city* is fine, but not, **destruction the city*. One familar explanation for this is that Verbs and Prepositions, but not Nouns, assign abstract Case, and that Case is required for an element to be "pronounced," in effect. Thus, in order to realize *destruction the city*, one must insert a dummy preposition so that *the city* receives Case and can surface: *destruction of the city* Note that adjacency of a surface NP immediately adjacent to V's and Prepositions element could be used as evidence that Verbs and Prepositions, but not Nouns, assign case.

19. Lexical categories are actually defined by bundles of (binary-valued) distinctive features; these will be described shortly.

20. The exceptions involve coordinate constructions (not considered in this thesis); category conversion rules that alter the syntactic class of a phrase (see Jackendoff [1977 pages 51-53]), discussed later in this chapter; and (possibly) a variety of phrase structure rules that introduce "Sentence" phrases (in other languages; see Lapointe [1980]).

$$*X^{n} \Rightarrow \ldots X^{n-1} \ldots X^{n-1}$$

According to these definitions the category on the left-hand side of phrase structure rules must possess the defining category features of the Head on the right-hand side. And, because the features of Head categories are themselves ultimately projected from those of actual lexical items, the phrase structure rules themselves are "projections" of the features of lexical items. It is this constraint which is meant to capture our intuitions that whatever a *Noun Phrase* is, it is chiefly based on the notion of *Noun*.

The remaining components of an X-bar phrase structure schema are the flanking $\underline{C}_i$, representing items that can precede or follow the Head of the phrase. These elements are either the so-called Specifier of the Head, an item or string of items that might be thought of as "particularizing" the head category, as *the* does in *the ball*; or the Complement of the Head, a category that, intuitively, forms a part of the predicate-argument structure of the verb, "complementing" it.

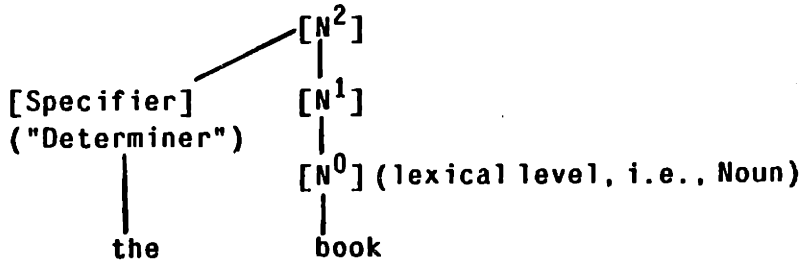
Taken together, the set of possible lexical categories plus the number of "bar levels" determines the total inventory of possible syntactic categories in a base phrase structure component. The aim of the X-bar theory is therefore to distinguish systems of human phrase structure rules from the string-to-bracket mappings describable by an *arbitrary* context-free grammar. It does this by judging as ill-formed many *a priori* possible trees, narrowing the characterization of human phrase structure rules. For example, the first structure below, but not the second, is a valid phrase structure tree according to the conventions just presented:[21]

---

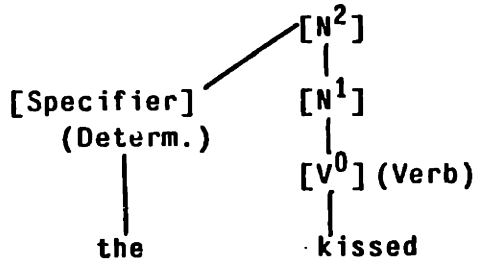
21. Some readers may have noticed that the restrictions given so far ban the occurrence of two immediately adjacent Nouns or Verbs in surface structure. But there are many English phrases where two nouns do occur adjacently, e.g., *baby doll*; *garden path* -- cases of so-called *Noun-Noun* modification. Here, one noun serves in role of an adjective (a modifier) for the other. The X-bar theory as outlined in this section does not deal with these complications.

However, these items do not really form higher-level phrases (e.g., phrases at a one-or two-bar level). They function as single words, that is, as Nouns. Therefore it might be expected that their analysis and acquisition would lie outside the X-bar system *per se*. Note that the analysis and acquisition of these words could draw on the X-bar system nonetheless. Some kind of mechanism for such "word syntax" is apparently required anyway, to account for the process of compounding to form words like *blackboard* or *redneck*. Though word and phrasal syntax are distinct levels of representation, they interact (one feeds the other) and (probably) use tree structures and formal devices of roughly the same power (but not of the same theoretical vocabulary). The formal convergence between word and phrasal trees provides additional support for both levels of representation, as well as an explanation for apparent phrasal-syntactic bias in the Noun-Noun modification examples.

The existence of compounding rules suggests that there is some process that modifies the category membership of a Noun such as *garden* so that it can be fit as an Adjective. As outlined later in this chapter, a feature copying mechanism adopted from Williams' [1981b] approach accounts for many such phenomena: in this account, given *garden path* as a Noun, *garden* is labelled "Modifier" in a top-down fashion due to independently-motivated feature copying processes in English that take *path* as a Noun.



(a) Permitted phrase structure tree.



(b) Prohibited phrase structure tree.

Figure 2.2 - Permitted and prohibited phrase structure for Noun Phrases under the X-bar System.

The effect of the X-bar theory is to rule out the tree structure in Part (b) of Figure 2.4 in favor of structure (a).

Perhaps more importantly, the X-bar system demonstrates that there is a tremendous redundancy in possible human phrase structure systems, a redundancy that is not captured by the context-free re-write notation. There are no natural phrase structure systems where the constituent order is of the form Head-Complement for some phrase types, and Complement-Head for others; once the order is set for one phrase, say, Verb Phrases, then it is the same across all phrases. In a sense, there is no phrase structure grammar at all, but merely the order Head-Complement, plus a selection of possible Heads (Nouns, Verbs, Prepositions, Adjectives). Of course, a context-free re-write system can perfectly well *describe* any set of possible constituent structures -- since the X system is a restriction of a general context-free grammar. But this fact is on a par with the fact that a Turing machine program notation can describe any possible computable function: a fact that is true, but unenlightening. Unless further, principled restrictions are advanced, there would seem to be no way to *explain* this regularity in phrase structure systems. Since the minimal fact that demands explanation is that Head-Complement order is generally fixed across all categories of type X, where X=Noun, Verb,

Preposition, Adjective, any explanation would seem to require at least the outlines of a proposal like X theory.

This redundancy is also an aid to acquisition. It reduces the amount of information from the external world that must be used in order to "fix" the right grammar. This property of the X system is analyzed from a formal point of view in Chapter 3. One can regard the X system as providing a finite stock of "parameters" that specifies the possible class of natural phrase structure systems. One such parameter is whether Complements attach to the right or left of heads; the learner's task is to set the values of those parameters correctly. It is a fact of English phrase structure, for example, that the Complement Noun Phrase Objects of Verbs attach to the right of the Verb Head (*Sue kissed Mitch*); in Japanese the reverse is true.[22]

Taken as an acquisition model, the X-bar theory reduces the task of inducing the right phrase structure rules for a particular language to a series of "parameter-setting" decisions about how to assemble the raw materials provided by the X-bar theory into new phrase structure rules.[23]

An X-bar theory must make several decisions about form and labelling possibilities, roughly, decisions about tree shapes and node labels. Some of these are fixed; other design decisions represent what the open parameters of the system are. Let us review both types here.

Form of Phrase Structure Trees

---

22. One might wonder just how strong the X-bar constraints really are. It well known that a context-free grammar can be represented by the set of (labelled) trees that it generates. Therefore, some set of form (tree shape) and labelling (node labels) parameters should suffice to completely specify *any* set of context-free phrase structure rules. Do the X-bar conventions actually constrain the class of possible context-free grammars at all? Consider an intermediate result in establishing Chomsky normal form for cfg's: every context-free grammar can be written as A-->a (a is a terminal); or A--> $B_1B_2...B_m$, for m greater than 1. But m must be unbounded for this approach to work. To pass from this result to Chomsky normal form, A-->BC or A-->a, "flat" phrase structure rules of length m must be re-written as a tree (with distinct indices on all nodes) of depth m-1 (at least). Thus, in order for the X-bar schemas to account for all context-free grammars, either the number of "bar levels" or the number of distinct categories must be unbounded. Since the number of levels in an X-bar system is most often restricted to three or so, the first alternative is ruled out. If one assumes a finite distinctive feature system for categories, so is the second. All that this indicates is that constraint in the X-bar system flows from the restrictions on categories and levels, rather than from a limitation to ternary or binary-branching alone (as is immediately apparent from the Chomsky normal-form theorem).

What is not so well known is that the node labels themselves are mathematically superfluous: Joshi and Levy [1977a] have shown that every context-free grammar is weakly equivalent to some grammar representable as a set of unlabelled trees - at the cost of an enormous expansion in the number of rules required. Apparently, although node (category) labels are not formally necessary, they serve to cluster phrase structure rule systems in such a way as to reduce the burden of storage and acquisition: once again, the depth of the resulting trees (as well as their number) would be enormous.

See Chapter 3 for additional discussion of tree automata induction and the place of non-terminal labels in grammars.

23. An important issue is whether there is an ordering to the triggering evidence that must be supplied for setting the parameters. In some cases, ordering can further reduce the amount of information that is required to fix a rule system; Chapter 3 works out this possibility in the case of a phonological rule system, based on the work of Kean [1974].

(1) The maximum number of X-bar levels.

The number of levels of phrase structure permitted will, in general, determine the possible hierarchical arrangements of constituents in a phrase. For example, if no levels were permitted (that is, the maximum index of any schema were zero), then the resulting phrase structure "trees" would be flattened to a simple linear ordering of their respective lexical parts. A "Sentence Phrase" would reduce to, roughly, a string of words, and there would be no syntactic structure, beyond that available in the precede-and-follow sequencing of words (or, more precisely, category labels for words):

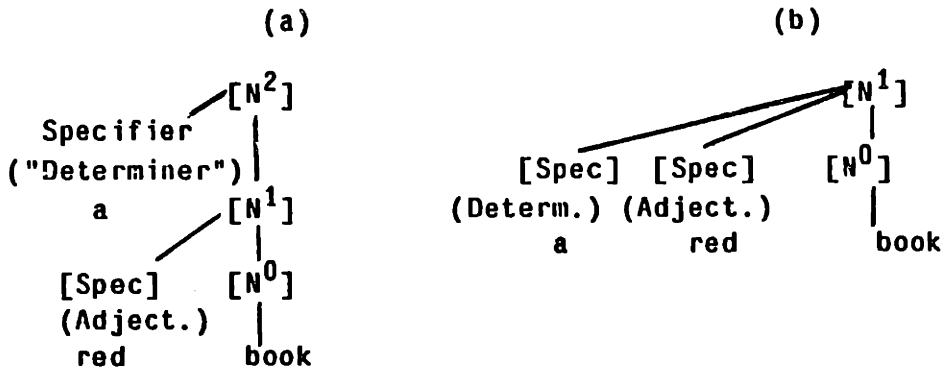
$N^0V^0N^0$ (Noun--Verb--Noun)

But as is obvious, a simple listing of possible word sequences cannot account for the full recursive generative capacity of human grammars. Recursive rule schemas are evidently required in some part of human grammars, and this power cannot be supplied by non-recursive, flat phrase structure schemas. Since the phrase structure system is intended to be the repository for the basic hierarchical structures allowed by a grammar, it seems natural to place recursive power there.[24] The other powerful and well-known argument for the existence of *phrases* is, of course, the existence of ambiguous sentences. Individual words must be grouped as units, otherwise, as Chomsky pointed out [1955 page 177], *They are flying planes* cannot be ambiguous (as either [are flying] [planes] or as [are] [flying planes]).

This means that the maximum number of levels allowed by the X-bar schema must certainly be greater than zero. How much greater? The same sort of phrasal-grouping arguments can be advanced to show that the maximum exponent in the X-bar system should be at least two. Without going into extensive justification of this result, it is enough to consider that under one interpretation of English Noun Phrases such as *the red book* or *two red books*, the Determiner *the* or *two* clearly modifies the pair of words, *red book(s)*. Assuming a principle of semantic transparency, this semantical grouping is posited to have a syntactic reflex; hence there must be a distinct node bracketing the Adjective and Noun together. That is, tree structures of the form (a) below, as opposed to those of form (b), are permitted:

---

24. Note that in early generative accounts (Chomsky's *The Logical Structure of Linguistic Theory* [1955]), this power was placed in a transformational component.



Though a Head+Complement relationship is apparently necessary to properly describe the clustering relations of the constituents of phrases, it is still not clear whether any additional, higher-order constructs are required; the specification of an X-bar theory with the "right" number of levels has been and continues to be a point of some controversy. Some authors, like Jackendoff, advance a three-level system, based upon arguments like that given for the move from one to two levels of structure. In this thesis, the following system will be adopted: an X^0 level, corresponding to an actual lexical item; and an X-max level, corresponding to NPs, VPs, PPs, or APs. Additional levels are constructed *as required*; for details, see the discussion on $\overline{X}$ parameters, under the heading of "Level number for Specifier (Complement) attachment." In brief, a new non-terminal category is created whenever there is evidence that two phrases are *not* in the same equivalence class with respect to parser behavior. For example, NP and $\overline{S}$ alternate as the Complement to *want*: *I want John to go* -- *I want the candy*, and so in this case NP and $\overline{S}$ are attached to the same bar-level of the VP. But a phrase that does not serve as the argument to a V, as *very badly* in *I want John to go very badly*, must be attached to a different bar-level from that of the NP argument *John*. So far, two levels of complement structure have proved to be sufficient, one level for argument attachment to the lexical Heads of phrases, and one level for attachment to the maximal projection as a whole.[25]

(2) Uniqueness of Heads.

Except for coordinate structures not covered in this report (e.g., conjunction), there is at most one Head on the right-hand side of each phrase structure rule. It follows that there can be no phrase structure rule of the form $X^n \Rightarrow X^{n-1}\ X^{n-1}$, e.g., NP⇒Determiner Noun Noun. This is a fixed requirement, not a parameter that is set via positive examples.

(3) The Head of Sentence Category

---

25. Jackendoff [1977] adopts a "uniform level" hypothesis: all phrases have the same number of levels of structure, three. This constraint may be too strong. The acquisition procedure avoids this problem by interpolating new levels as demanded by the requirement that equivalence classes of non-terminal elements be identified with distinct bar-levels.

As described so far, the X-bar conventions fail to account for the existence of Sentence (or "S") constituents in phrase structure. There are basically two options to deal with the exceptional behavior of Sentences; both have appeared in the literature: (1) S lies outside some of the X-bar Conventions. In particular, S has no lexical Head (i.e., only S^1, S^2, S^3 categories). On this account, S could be introduced by rules of the form $S \Rightarrow \{N^3, V^3\}$.[26] (2) (i) S abides by all the X-bar conventions described previously. In particular, V^2 (e.g., a Verb Phrase) is the Head of S, and, furthermore, S = Vmax;[27]
(ii) as in (2i), but with INFL (the inflection element) the Head of S.

There are arguments for and against each of these possibilities. The first alternative offers flexibility: with special rules for introducing "S" phrases, one can directly account for a greater variety of S constituents and idiosyncratic behavior. Its (possible) cost is an expansion of the set of phrase structure schemas. The second approach has nearly dual advantages and disadvantages: only one schema is used for acquisition, but this may not suffice for all human phrase structure systems; in addition, one may need special "S" rules anyway, for co-ordinate constructions (*Sally kissed John and then Bill kissed Sally*). The third choice maintains the uniformity of the operator-operand structure implicit in the Head-Complement structure. This report has opted for the third alternative, with INFL as the Head of S. Note that this another example of a fixed requirement that is not acquired via parameter setting.

Finally, there is a semantical connection between the structure of sentences and what is assumed known about semantic structure that is crucial to the success of the acquisition procedure. A sentence is taken to be, roughly, a proposition consisting of a Subject and a Predicate. The Subject may be thought of as the NP that, when composed with a predicate, yields a complete proposition. A predicate itself consists of a verb plus its arguments. So for example, in the sentence, *John kissed Bill*, the predicate corresponds to the VP *kissed Bill*; it is constructed out of the V *kiss* and the NP argument *Bill*. (*kiss* could be considered to be a function with a domain specified by the argument *Sally*; the output of this function is the predicate *kiss Sally*.) Crucially, Complements are assumed to be arguments in this sense. *John* is the Subject of this example; it is composed with the predicate *kiss Bill* to yield the desired full proposition. How should this be represented? One way is via the lambda-calculus notation suggested by Williams [1977]. The predicate form becomes a lambda-body,

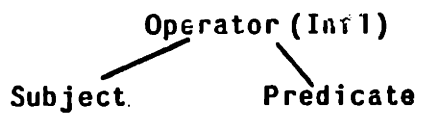
---

26. See Chomsky [1970]; Hornstein [1977]; LaPointe [1980].
27. See Jackendoff [1977]; Marantz [1980]. This approach does not handle grammars with apparently "unspecified" category rules, e.g. $(X^0)^*$ expansions as in Dyribal or Warlpiri. Nor does it deal with Sentences apparently lacking verbs. See Lapointe 1980. Note that, intuitively at least, the S = Vmax approach implies that Sentences are ultimately verb-based, or, using an ever looser terminology, "event-centered."

with the Subject being abstracted. So for example, *John kissed Sally* is represented as,[28]

λx [x kissed-Sally] John

This semantical analysis has a syntactic counterpart. S's (full propositions) are assumed to be composed of of three parts: NP, INFL (an inflectional element), and a VP. The NP corresponds to the logical Subject of a Proposition, the VP to the predicate, and INFL, to a tense "operator". On this analysis, a proposition is composed of a logical Subject and a predicate bound together by the INFL operator:[29]



Thus we have Infl: [Subject] x [Predicate] --> an object of type [Proposition].

In general, all maximal projections have this basic operator-operand form, but the category types differ depending upon the type of the operator. For example, consider Noun Phrases. Intuitively, they denote objects. Thus the output type is most naturally thought of as an expression denoting an object, in some sense, rather than a proposition or a predicate. Similarly, Verb Phrases consist of Operators (verbs) that form predicates out of arguments that have Case (either via previous case assignment or by means of adjacency to the operator) and Prepositional phrases are operators that output NPs bearing out of Case-less NPs.

---

28. For a more recent analysis along these lines, see Marantz [1981]. As Marantz notes, there is evidence for an abstraction on the Subject rather than the Object, because changes in the content of the Object can change its semantic role, whereas the semantic role of the object seems to be fixed:

throw a baseball
throw a party
throw support behind a candidate

vs.

NP threw
throw NP! (Imperative)
etc.
(examples from Marantz)

29. Note that the logical Subject need not be the grammatical Subject. This also parallels Marantz' analysis.

WH elements also act as operators, and are non-arguments: e.g., *Who did Sally kiss?*.

Arguments are also "complete" semantical units, in the sense that one can assign a semantic interpretation to such objects locally, without looking at the remainder of the sentence. So for example, *Bill, to school,* or *Bill silly* (in the small clause, *I tickled Bill silly* are all arguments. In contrast, *kissed* or *the boy* in *the boy with red hair* are not arguments, since they are not complete semantical units. Evidently, this is not sufficient to define the notion "argument," since intransitive verbs are also complete in this sense but are not arguments. In addition to semantic completeness then, arguments possess the additional property of being assigned a thematic role -- as Agent, Affected Object, Goal, and the like.

Note that we have *not* said that the syntactic nodes corresponding to these types are necessarily distinct. For example, the INFL and VP nodes could be identified with one and the same literal syntactic object, namely, a projection that includes both an inflection operator and a predicate. This will turn out to be possible just when no evidence has been received that calls for the VP and INFL nodes to be "split," i.e., in just that stage of development where lexicalized proxies for INFL such as *will* are unknown. In other words, there will be a certain stage where the analysis of sentences is, NP INFL/VP, where INFL/VP is an abstract category that combines both aspects of tense and predication. In $\bar{X}$ terms, this means that the projection of whatever underlying lexical token it is that forms the basis of the phrase has not yet been completely distinguished from other lexical projections.

This notion corresponds precisely to the notion of "equivalent state" from automata theory; for a formal analysis, see Chapter 3. Later evidence will prompt the splitting of the INFL/VP node into two parts, the more traditional INFL and VP nodes. The existence of "abstract" nodes of this sort is not surprising; for example, Aoun [1979] has suggested that discontinuous Verb and NP arguments form an "abstract" VP node in languages such as Arabic.

Given this analysis, then there will be a tendency to assume that the first NP in a sentence is the "logical Subject" of a sentence (proposition), in accordance with the constraint that all sentences are of the form NP-Predicate. Note that the thematic role of this NP is undetermined; it could be either the Agent (in accusative languages such as English), or the Affected Object (in ergative languages -- is this correct?). Note that this is a that is in fact observed; early sentences understood and produced by children are of a simple NP Subject - Verb - NP Object type. Even in languages where case marking permits flexibility in word order (e.g., Japanese, Latin), there is a tendency to place the Subject before the Object.[30] There will also be a tendency to take the second NP as part of the complement (argument) structure of the the predicate. In short, by assuming that propositions are of the form NP

---

30. SOV, VSO, SVO, but not OSV, VOS, OVS.

See J. Greenberg [1963].

INFL VP, (perhaps NP INFL/VP) and that this NP serves as a logical Subject (and not a Complement to an operator of some sort), one can use surface constituent order to determine whether the language at hand is of the form Head-Complement or Complement-Head. So for example, the surface order NP NP V will be taken to be of the form: (logical) Subject - Argument to Predicate (Complement of Predicate) - Verb (Head of Predicate). Hence it may be concluded in this case that Complements precede Heads in this language. Similarly, the order NP (logical Subject) - Verb (Head of Predicate) - NP (Complement of Predicate) indicates a Head-Complement order. Plainly, this approach assumes that an acquisition procedure has access to the "correct" predication relations of a given sentence, at least for some early examples. This method is illustrated in example acquisition scenarios below.

Labels of Phrases

(1) Heads: Inventory of (lexical) categories for Heads.

As mentioned, since the Head of a phrase is a projection of a possible lexical category, the inventory of such categories will delimit the the range of possible phrases in a system of phrase structure rules. For example, if the only distinguishable (i.e., categorizable) lexical items were "nouns", then the only permissible phrases would be Noun Phrases. Thus, a theory of "possible lexical item" is prior to a complete formulation of this part of the X-bar system. If this is so, then this is one place of critical contact between the acquisition of individual lexical items (and their features) and the acquisition of syntactic (phrase structure) rules. The X-bar restrictions may provide a set of skeleton phrase structure trees, but without a proper set of (lexically derived) features to label the nodes of the trees, this work has been in vain. In short, a complete understanding of the acquisition lexically based phrase structure must rest on an understanding of the acquisition of words.

Whatever other "features" words may have that permit their categorization, one fact stands out: all languages have some items that can be dubbed "nouns" and others that can be called "verbs." Suppose these two classes are taken as primitive (for whatever reason); that is, assume that the Lparsifal procedure can initially distinguish a lexical item as of the class [Noun], the class [Verb], "neither", or both.[31] As described earlier, it has often been supposed in the literature that this categorization could be effected by some kind of semantic grounding, in the simplest case by identifying the class [Noun] with the "semantic" class [Object], and [Verb] with [Action]. Early examples use this grounding to draw a semantic-syntactic correspondence.[32]

If these two classes are interpreted as binary-valued attributes, ±Noun or Verb, there are, of course, actually four possible lexical categories, hence four possible phrase types: [+Noun -Verb], [+Noun

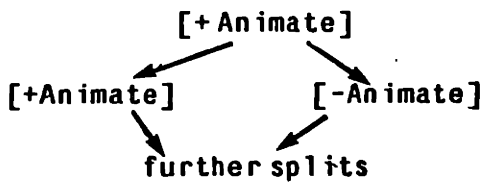
---

31. In other words, ambiguous cases are simply left unmarked.

32. To repeat an early discussion, this approach once again presumes that there is initially a close, possibly one-to-one correspondence between "syntactic" and "semantic" representations.

+Verb], [-Noun +Verb], [-Noun -Verb]. These four phrasal types constitute the Major phrasal categories: NPs, APs, VPs, and PPs.[33] Each of these items can form maximal projections -- they can be the Head of a phrase. Semantically, each can be an operator.

If we think of these four categories as actually bearing ontological weight, then it is possible to imagine a scenario for the development of these types. The idea is based on Keil's [1979] analysis of the development of ontological categories in children, discussed at length in Chapter 3. The basic notion is that new categories develop out of old ones via a process of foliation, that is, the splitting of old categories into new ones, rather than via any sort of radical reconstruction. Apparently, as Keil observes, children first develop a concept by determining what it is *not*, with respect to existing categories. For instance, it might be that a concept of inanimacy could be triggered by noting that there are objects that are not animate. This negative categorization is then "reified", becoming a category in its own right. This would lead to developmental pattern something like this:



Something very much like this process could go on in the formation of the major phrasal categories. Suppose the system starts with the categories [+Noun], [+Verb] corresponding to tokens that are "Substantive" (in a sense, denoting an object); and those that are "Predicative" (denoting actions, or, perhaps, qualities or states). Tokens that do *not* denote Substantive objects will be labelled [-Noun], and those that do *not* denote actions/qualities will be labelled [-Verb]. Prepositions will be formed as a category containing tokens that are neither Substantive nor Predicative. Adjectives are Predicative and also help pick out an individual object or action in a model-theoretic sense; i.e., they specify part of a denotation. Thus one arrives as the following simple developmental picture for the major categories:

---

33. This analysis basically follows that in Chomsky [1970].

```
      (Substantive)                                  (Predicative)
[+Noun]                                              [+Verb]

[+Noun]   [-Noun] (Not  Substantive)    [+Verb]   [-Verb]
                                                  (Not Predicative)
```

An extension to other categories (e.g., Particles) requires additional distinctive features; several have been suggested in the literature. Jackendoff [1977 page 32] proposes adding a feature that marks the ability of an item to take a Complement. For instance, English Noun Phrases can take Complements (e.g., Prepositional Phrase complements: *the boy behind the shed*) but Articles cannot (**the behind the shed*...). In fact, all four of the major categories of English allow phrasal Complements of one form or another: Noun Phrases (*the boy that I know*), Verb Phrases (*kissed Mitch*), Adjective Phrases (*green with envy*), and Prepositional Phrases (*off the wall*). (One might also put inflectional elements into this category, if they are thought of as having VPs as complements.) In contrast, the "minor" categories -- particles, articles, and the like -- do not seem to form operator-operand structures of this kind; they do not form phrases. Rather, they are incorporated into the maximal projection of some major phrase. Following this line of discussion, we use the feature ± Head to distinguish between these two types of elements. Note that every item must therefore either (1) be a Head, forming some maximal projection; or (2) be a -Head, a "bare" lexical token item that is attached as the Specifier or Complement of a maximal projection. (If Complements are always arguments, as is the case, then it follows that -Head items are always Specifiers. Therefore, we can use appearance as a Specifier to label a lexical item as -Head.) This gives us:

(2) Specifiers and Complements: Inventory of categories.
The items flanking a Head must themselves be either (1) Major Phrases (maximal projections) or (2) minor lexical items that cannot form phrases.

Form Parameters

(3) Left or right attachment of Specifier (Complement) to the Head.
This parameter was discussed earlier.

(4) Level number of Specifier (Complement) attachment.
With three levels, constituents can be attached at the X^0 (lexical), $\bar{X}$, or X-max levels. Jackendoff [1977] attempts to relate the level number to which a constituent is attached and certain semantic properties. For instance, he reserves the X^1 level for constituents that are subcategorized, e.g., the Noun Phrase arguments of verbs. In addition, Jackendoff highlights two other types of complements: (1) restrictive modifiers, adding "extra truth conditions to the main assertion of the sentence"

[Jackendoff 1977, page 61], as in *John ran quickly to the store*; and (2) unrestrictive modifiers, like appositives, that contribute no extra conditions to the main assertion but rather repeat or emphasize it, as in, *John ran quickly to the store, of course* [Ibid., page 61]. The idea is that restrictive complements are somehow "bound" less tightly to the Head category than functional arguments, but more tightly than appositive-like items. Therefore, Jackendoff claims, if functional arguments are attached at the X^1 level then restrictive modifiers should go at the X^2 level and unrestrictive modifiers at a top-most X^3 level.[34]

How then is an acquisition procedure (or language learner) to distinguish among the three sorts of complement functions? One straightforward possibility is to assume that the learner can pick up the functional differences on the basis "semantic" context. Obligatory subcategorization could correspond to the observation that a certain complement type "picks out" different objects from a set of individuals and is never absent (e.g., *kiss someone* but never *kiss* ____). Restrictive modifiers could be flagged as complements that can aid in denotation, but are sometimes absent (e.g., *kissed John behind the car*). The remaining sort of complement, neither obligatorily observed nor apparently crucial for denotation, could be assigned to the non-restrictive category. Thus, this would be another -- and uncontroversial -- situation where "semantic" knowledge -- contextual, pragmatic, or conceptual cues -- might be brought be bear in the acquisition of lexical meaning.

Jackendoff's proposal for determining Complement attachment is a one-for-one correspondence between semantic (functional) distinctions and syntactic structures (at least in English). What happens with Specifiers? Here, as Jackendoff admits, the regularities are not so plain:

> Specifier systems involve very small numbers of lexical items and are riddled with idiosyncrasies. Thus general phrase structure rules must be supported on the basis of impoverished and skewed surface distributions.
> [1977, page 103]

This observation clearly bodes ill for an acquisition procedure; if a linguist cannot draw order out of a large corpus of data, then what of a child? If we regard the X-bar conventions as a reflection of constraints on acquisition, this conclusion is not at all surprising: it says that this particular portion of the X-bar system need not be acquired in a parameter-driven fashion by via triggering on semantic-syntactic regularities, but rather by (largely) brute-force, example-driven induction techniques that pay attention to idiosyncrasies in the data. (And only within a restricted domain: the level of the specifier attachment, not the left or right attachment to the head.) There is in fact some

---

34. Pinker [1980] also adopts Jackendoff's scheme in his model of acquisition: the idea of equating bar-levels with "semantic" abilities for the purposes of acquisition is taken from his report. Details of children's acquisition of restrictive and non-restrictive clauses seem to be lacking.

evidence from the acquisition literature that would support this view.

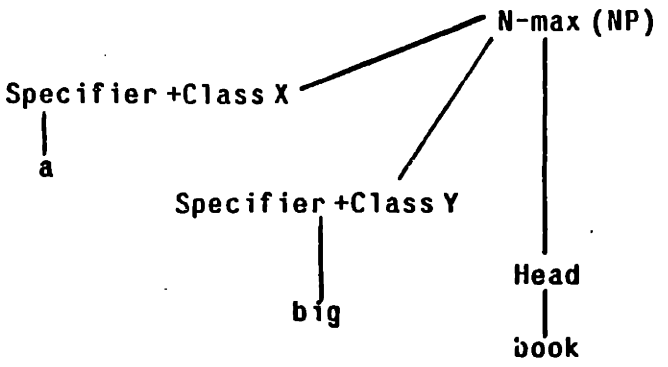
What does this possibility amount to in practice? Separate lexical categories attach to separate levels of the X-bar template (this is simply a re-statement of the definition of category and the form of the X-bar template). For example, if it can be deduced that *the* and *red* are separate categories, then they must be placed on two different Specifier bar levels, corresponding to distinct non-terminal elements. How might this be determined? Some examples will show that *red* and *the* should be placed in the same class, e.g., *Red books are expensive/ The books are expensive.* Other examples show that they must be in different classes, however: *The red books are expensive* shows that the left-hand context of *red* can contain *the*, whereas *the*'s left-hand context cannot. More generally, this approach is simply the familiar one of the induction of a finite-state automaton, now broadened to the case of the general Parsifal interpreter.[35]

That is, one can do away with notion of different bar levels altogether if one simply adopts the view that bar levels just represent different states of an automaton. Then new states can be formed out of old ones on demand, without any reference to bar levels. Formally, this procedure is just that used in the induction of finite state automata, and is reviewed in more detail in Chapter 3. The basic idea is that a non-terminal category is split into two new ones when positive evidence is received that two elements do not "behave alike." So for example, since *the* and *red* do not "behave alike" -- e.g., *red* can be followed by a complement phrase, but *the* cannot be. This means that this items must be members of a different equivalence class of tokens. If the grammar is to represent these differently, then it must associate two different non-terminal labels with these two elements. Thus, the examples, *a book, the book, a red book, the red book* are sufficient to form the right classes. *A* and *the* have the same suffixes, hence go in the same states. (A later example will split these two, since *the books* is grammatical but *a book* is not.) *Red* and *the* are not in the same class, because there are no full grammatical examples of the form, *red red book*, or *red book*. Therefore, the classes {red} and {the, a} must be different, and they are attached as different Specifier nodes. This labelling is carried out automatically by creating separate Specifier nodes and annotating them with the class name of the node attached. For instance, the Noun Phrase, *a big book* will be labelled as follows:

---

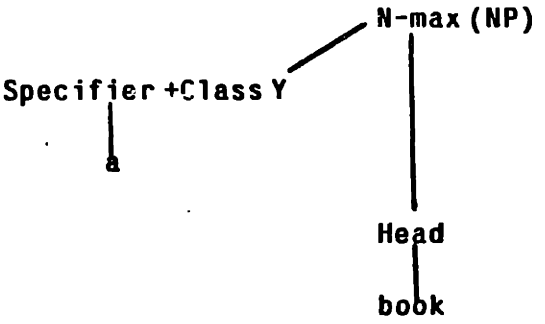
35. Note that for this approach to succeed with just positive evidence, the system must know when all possible transitions of the target finite state automaton have been presented via one or more examples; otherwise, it cannot possibly induce the proper target machine.

Chapter 3 analyzes the complexity of this approach. It is well-known (see Gold [1978]) that the induction of a minimal finite-state machine from examples is NP-complete in the number of examples that must be examined. However, in the case of a state merger-and-refinement approach that uses its data carefully, only a polynomial number of data samples need be used; see Angluin [1982].



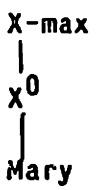
Class Y= {a, the, ...}
Class X= {big, red,...}

There are then two ways of distinguishing equivalence classes of the parsing machine: (i) by different non-terminal labels, as projected from different lexical equivalence classes; and (ii) by different attachment points, as given by Specifier level numbers. The second method is illustrated by the example above; the first is illustrated by the following example:

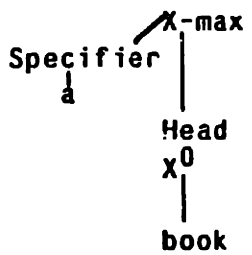


Class Y= {a, some,...}
Class Z= {the,...}

Developmentally, category splitting follows the refinement procedure suggested earlier; that is, new categories are not formed until needed. Thus if the only examples of NPs seen are names, like *John* or *Mary*, then the X category developed would look like this:



An example such as *a book* leads to:



But what about a phrase like, *the big red book*? It is well known that there are at least two possible bracketings corresponding to this surface string: [*the* [*big*[*red book*]]] or [*the* [[*big*][*red*]][*book*]]. The approach sketched above offers no way to discriminate between this pair. *Red* and *Big* are presumably both known to be members of a single category (they occur in identical positive contexts, *the big book*; *the red book*, ...). Given this piece of evidence, the following protocol is applied: if two lexical items of identical category membership occur adjacently in surface structure, then their phrase structure expansion is collapsed via the use of the Kleene star notation $(X)^*$. (See immediately below for discussion of this point, which is intimately connected to the *non-counting* property of natural languages.) Thus *big red book* would by default become [[big red][book]], with a basic Noun Phrase expansion like Article (Adjective)* Noun.[36]

(5) Optional, obligatory, or repeated constituents.

By default, it is assumed that constituents are obligatory until proven otherwise. This result follows from the Subset Principle, discussed at length in Chapter 3. Plainly, assuming that arguments to verbs

---

36. Of course, this does not preclude the development of a separate theory of internal Noun structure; it just means that such a theory becomes a sub-domain of a general theory of lexical acquisition. Because Noun structure must ultimately comport with phrase-level structure, one might even expect syntactic structure to play an important role in such a theory; it also seems likely that the well-formedness of internal Noun structure will turn on the regularities of as yet ill-defined "semantic" properties, as well as a theory of lexical acquisition. For an initial approach, see Marcus [1980, Appendix A]. Incidentally, if the proper theory of the lexicon is like that of Lieber [1980] – who adopts unlabelled, binary-branching trees as the basic syntactic form of lexical items – then the similarity between word-level syntax (hence Noun-Noun groups) and phrase-level syntax is in some sense automatically accounted for – both would indeed use the same sorts of structures. This would also explain Marcus' observation of syntactic bias in Noun-Noun modification.

are obligatory generates a smaller set of surface strings than one assumes that the constituents may be obligatory or optional. If a procedure first guessed that a restrictive complement was optional, it would never receive disconfirming evidence; there would be no way for a procedure to recover from a faulty hypothesis of optionality. On the other hand, by guessing a more restricted language (namely, one where all arguments are obligatory), it could protect itself: if the procedure finds itself to be wrong (it receives a piece of input data where the item is not present) it can still retreat to the broader position of optionality.[37]

Repetition of Constituents

In accordance with the Subset Principle, the procedure acquisition will not permit the repetition of a constituent in phrase structure unless a positive surface string indicates it. But if any repetition is observed, the procedure permits arbitrary repetition (in apparent violation of Subset Principle). The choice is whether an item of a given type allows an (indefinite) number of immediately adjacent elements of the same category in surface form -- in the Kleene notation, a choice between X and X^*. For example, English Adjectives and Prepositional Phrases can be indefinitely repeated: *big red ...ball*; *behind the table in front of the chair....* But not so Articles Adverbs, or Modals: **the a book*; **kissed quickly softly*; **I can will kiss...*). Why is this choice an appropriate one?

In brief, the system is designed so that we cannot state in a phrase structure rule that, say, a Prepositional Phrase can be repeated exactly three times. According to this approach, phrase structure rules cannot "count". This observations seems to be descriptively correct, for most (perhaps all) linguistic phenomena. Is there an explanation for it? Speculating, the reason why (human) grammars apparently obey this property may actually run quite deep: counter-free parenthesis grammars seem to be more easily inducible from sample evidence than counting automata (fCrespi-Reghizzi, Guida, Mandriolo [1978]; Crespi-Reghizzi [1971]). Observe that if natural languages (and grammars for those languages) are non-counting, then the "traditional" counter-examples demonstrating non-identifiability in the limit do not apply, because the natural languages will not include all finite languages.[38]

---

37. This fact has also been noted Williams [1981c].

38. Further, non-counting appear to characterize the structures that are locally testable. (There is a close correspondence between this formulation and the local transformation results of Joshi and Levy [1977].) Thus, the non-existence of counting phrase structure rules may in fact reflect their relative un-learnability (low ranking by the evaluation metric), a fact perhaps connected to the non-local power required for their computation.

2.4.4 Computer Implementation of the X-bar Rules in Lparsifal

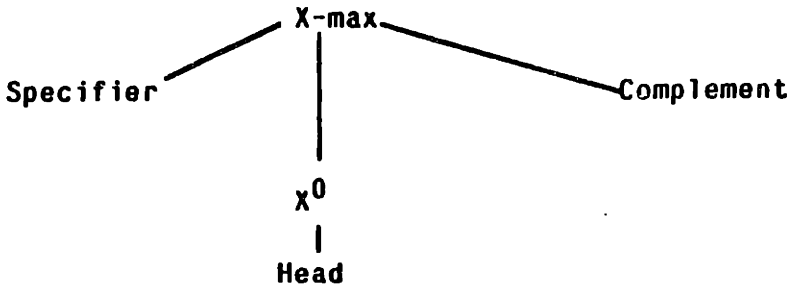
In order for the X-bar theory to be actually used by the acquisition procedure, several additional issues must be settled. The X-bar theory was not specifically designed as a process model for the parsing of context-free base rules; as a result, the remaining implementation details center upon the interaction of the static X-bar Conventions with the parser's computational activity. They are:

> (1) How should the X-bar template be encoded as a (fixed) data structure for program control?
>
> (2) What protocol should be established to ensure the proper creation of new X-bar nodes, their placement on the active node stack, and their completion (signalled by dropping the phrasal node into the buffer)?
>
> (3) What house-keeping activities must take place to initiate parsing? In particular, what is the place of S (or S-bar) node in the X-bar framework?

Each of these points is covered below in turn.

2.4.4.1 Encoding the X-bar Trees

As described briefly above, the on-and-off switching of sets of grammar rules is carried out by associating a bundle of grammar rules (a packet) with one or more of the components of the base (phrase structure) rules of a generative grammar. This is carried out by literally pairing a packet name with the fringe of an X-bar schema:



Of course, these packet names are unordered to begin with; that is, one starts with simply the set {Head, Specifier, Complement}. The packet name associated with this particular projection is determined by the identity of the Head lexical item. For example, if the Head is a Verb [+V -N], then the associated packet names are labelled as V's: Parse-Specifier-V; Parse-Head-V; and Parse-Complement-V. Actually, the packets are labelled with the feature bundle that stands for the label "V," namely, [+V -N]. In fact this bundle of features denotes an equivalence class that is *acquired*, namely, it is the class that consists of all those lexical items that behave just like V's do. Here, the definition of "behave just like" means "can be inserted in the same parsing context as."

The particular X-bar tree listed above imposes an ordering on the activation/deactivation of Parsifal grammar rule packets: after a schema is entered, packets are activated left-to-right as specified by the tree. In the simplified skeleton tree above, this would mean that if the X schema were entered,[39] the packet associated with the "Specifier" portion of the tree -- presumably, a packet containing grammar rules to parse Specifier constituents of whatever particular X-type phrase has been instantitated -- would be activated. After a complete constituent is built (or skipped, if this portion of the phrase is optional), then this packet de-activated and the next packet in line, Parse-Head-X, would be activated. The Head is an obligatory constituent, consisting of a single lexical "operator", like a Verb, a Noun, or a Preposition. After the Head is attached to the parse tree under construction, the Head packet is deactivated, and the Complement packet activated. (Sentence phrases are treated specially; see below.) Entry into a packet is used to drive the creation of new phrases as well, as discussed immediately below.

What happens when we come to the end of the instantiation of a X schema? This means that an X-max phrase must have been completely built. Observe that if the acquisition procedure is provided with the basic predications involved in a sentence -- the operators and their arguments

---

39. Recall that this triggered by the presence of items in the input stream, i.e., in a data-driven fashion.

-- then it will have substantial information about whether this phrase is in fact completed or not, because of the close correspondence between the syntactic level of representation and the predicate-argument level of representation. So for example, given the sentence, *John kissed Sally*, suppose that *John* is known as the logical subject, or even simply the Agent, of the main proposition, and that *kissed* denotes the basic Predicate or Action. In effect, this information says that *kissed* incorporates its NP arguments, rather than the other way around. But then, *kissed* cannot be incorporated into the phrase that dominates *John*; that would be an invalid semantical structure.[40] Similarly, if the end-of-sentence marker # is encountered whatever constituent is being built must be completed. Thus the end-of-sentence indicator acts as a helpful guide to unambiguously fix the right end of constituents.

If all arguments to an X-max have been analyzed, then all is well; the parse simply continues by dropping the completed X-constituent into the input buffer for attachment. Crucially, this operation takes precedence even over grammar rule failures, that is, the X-max is dropped if known to be completed, even when the acquisition procedure has attempted to attach material to that phrase and has failed. For instance, given a sentence such as *John kissed Sally* with *John* known to be a complete semantical unit, the system will attempt to attach *kiss* as part of the N-max *John*, fail because of the knowledge that *kiss* is not an operand of *John*, and, failing at both *switch* and *drop trace*, continue with the parse by dropping the X-max into the buffer. In short, failure of the three attempted actions is *not* considered a total failure when the X-max under construction is known to be complete; rather, the completed phrase is simply inserted into the buffer and the parse continues. This procedure simply has the effect of making a Complement (or Specifier) optional if a constituent is semantically complete without that Complement or Specifier. Note that the notion of semantic completeness is given *a priori* is simple cases, via the provided semantic representation (thematic structure).[41]

In more detail, completion of the tree will pop the "X" as the top node in the Parsifal current active node stack and uncover the next node down in the stack as the new current active node, along with any packets associated with that node. The new constituent moves into first cell of the buffer, shoving the current occupants of the first through third buffer cells (if any) to the right by one position. The parse then proceeds.

What about Specifiers? Once the order Specifier-Head-Complement has been fixed, then

---

40. Note that it is more difficult to determine whether *kissed* dominates or is dominated by an NP in examples such as, *the girl kissed by Bill was my sister*. This is another example of how unambiguous cases make for easy acquisition.

41. Alternatively, one could simply drop an X-max node into the buffer when it is known to be complete, according to the provided "semantic" information. This would eliminate the need to change the definition of acquisition procedure failure.

material to the left of the Head is assumed to be part of the Specifier system of a phrase. To repeat a point brought up earlier, there are no non-terminal nodes associated with Specifier material as such. Instead, we simply set up different states as warranted by positive examples. Formally, different states may be defined simply as "different suffixes", where suffix is defined as the right-hand context provided by the buffer. A full analysis of this method, based on the notion of "state" from automata theory, is provided in Chapter 3.

Converting this system to a Lisp data structure is straightforward. To keep track of the current part of the schema under expansion, a special variable Ps-Pntr is set to the name corresponding to the appropriate part of the X-bar schema. Note that since Parsifal must suspend processing certain phrases (like Sentences) while it parses other phrases (like Noun Phrases), this method actually demands a stack of Ps-Pntrs, one for each pending X-max under construction. Finally, the mata-variable X is set to the category identity of the corresponding Head item (N, if a Noun, V, if a Verb, etc.; it is left as an X if the categorization of the head is unknown or ambiguous. [42]

The Lisp representation of the X-bar schema just mirrors the Specifier-Head-Complement structure order:

> Parse-X-max: (Parse-Specifier-X Parse-Head-X Parse-Complement-X)
>
> This particular list is ordered, since it represents the Spec-Head-Comp relationship of a particular language, as set by positive examples. Initially, the list is unordered.

Finally, the value of "X" is set to whatever particular features of the Head item are known. For instance, if the Head is known as [+N -V] (i.e., a Noun), then the value of X is also set to be [+N -V]. Thus, the identity of a maximal projection is determined by "percolation through the head," as suggested by Williams [1981b].

One special situation is worthy of note here. A sentence is assumed to consist of the set of elements {NP, INFL, VP} (initially unordered and, perhaps, with the INFL/VP aspects collapsed). Therefore, the X schema for sentences looks like this:

> Parse-INFL-max: (Parse-Specifier-INFL Parse-Head-INFL Parse-Complement-INFL)

---

42. This method resembles that of *skeletal parsing*, as described in Aho and Ullman [1972]. In this approach, often used for operator-precedence parsing, the identity of non-terminal labels is ignored, and just the basic phrasal structure recovered from the input string.

As mentioned earlier, initially, the features of INFL will be propagated from those of Verbs, via percolation; in effect, Sentences will be headed by elements with the features, [+V -N +Prop]. Where does the feature +Prop(osition) come from? By assumption, S's are taken to stand for full propositions, that is, expressions combining a logical subject and a predicate into some complete semantical unit. INFL will initially be defined as this particular cluster of features. Intuitively, INFL contains a tense operator, but this element has not yet been distinguished. The Specifier of INFL will be required to have a +N feature; the complement (assuming now a +V Head), will also have a +N feature.

2.4.4.2 Creating nodes

Recall that Parsifal constructs an annotated surface structure parse of an input string, connecting together several nodes referring to the phrase X-bar structure categories such as S, Noun Phrases, and the like. Since these nodes are not present in the input stream, the parser must have a way to create them, placing them for analysis on the active node stack. In the original Parsifal, the creation of nodes was handled by explicit commands in certain grammar rules that detected the unambiguous "leading edges" of constituents of a given type. For example, Noun Phrase node creation was triggered by an always-active grammar rule that executed when it detected a class of "Noun Phrase Start" words -- Determiners, Quantifiers, and so forth -- in the first (left-most) cell of the input buffer:

```
{RULE STARTNP in CPOOL
IF      1st item in the buffer is marked NPSTART
THEN    Create a new NP node;
        IF      first item is marked DETERMINER
  THEN  activate PARSE-DETERMINER
  ELSE  activate PARSE-QUANTIFIER-PHRASE1;
        activate NPOOL.}
```

The first portion of the action of this rule calls for the creation of a new Noun Phrase (NP) node. The remaining part of this rule's action deals with the packet activation feature of Parsifal (now replaced with the X-bar system). According to this rule, if the first item in the buffer is a Determiner (*the, a,...*), then the packet holding grammar rules to parse determiners is activated; otherwise, a packet to parse quantifiers is; in any case, a general-purpose Noun-pool packet is activated. Eventually (and if the phrase is grammatical), all roads would lead to the analysis of the Noun that must for the core of a Noun Phrase:

```
{RULE NOUN in PARSE-NOUN
IF      1st item in the buffer is marked Noun [+N -V]
THEN    Attach first item to a new NBAR node....}
```

creating a new N^1 node and attaching the actual lexical Noun to it. Thus, a completed Noun Phrase would look something like this:

```
 NP (N2 in X-bar terms)
  |
N-bar (N1)
  |
Noun (N0)
```

Similarly, the original Parsifal created Verb Phrases in a data-driven fashion upon the detection of a lexical item with the feature "Verb" in the input buffer (at an appropriate point in the parse, namely, after the analysis of any auxiliary verbs):

```
{RULE MAIN-VERB
IF 1st item in buffer is a Verb [-N +V]
THEN    Create a new VP node. ...}
```

And likewise for Prepositional Phrases:[43]

```
{RULE PP in CPOOL
IF 1st item in buffer is a Preposition [-N -V ...] AND
        2nd item in buffer is a Noun Phrase [+N -V +M ]3
THEN    Attach first item to a new PP node. ...}
```

In the revised Parsifal, all of these phrase-level node creations are to be handled automatically by the X-bar Conventions. Entry into a particular X-bar schema remains the same: it is triggered in a data-driven manner by the detection of certain items in the input buffer. But instead of having to list a node creation rule separately in each and every individual grammar rule where it may apply, we can exploit the abbreviatory power of the X-bar Conventions to substitute just one node creation protocol,

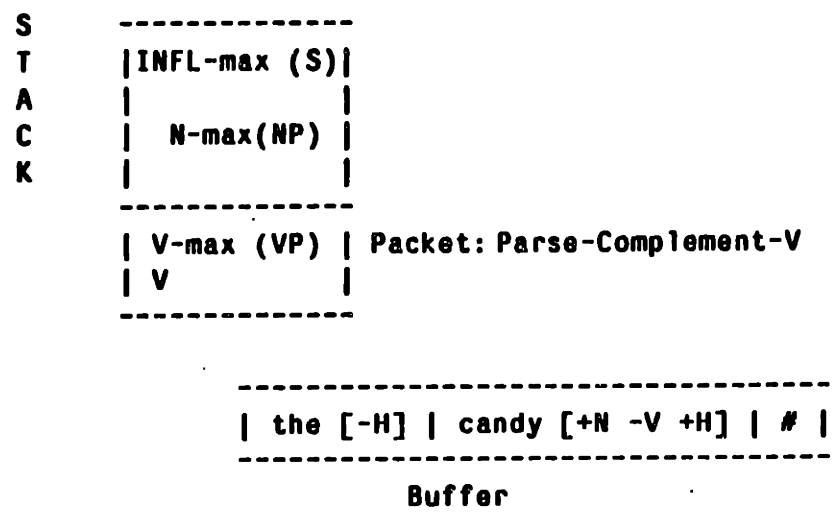
---

43. This is not exactly the rule in Marcus [1977, 1980]; this formulation has omitted the potential effects of *wh*-movement.

and make it part of the (fixed and universal) machinery of the interpreter. The creation protocol is simple to state: If a lexical item flags the start of a (new) X-bar schema of type "X", create an Xmax node of type "X" (making it the current active node), and enter the packet system for a schema of type "X". Thus, if a triggering item is of category [+N -V] (as an Article, Noun, or Proper Name must be), then an Xmax node of type [+N -V] would be created, and the packet schema entered. Similarly, if the item is marked [-N +V] (a Verb, say) then parsing of a an Xmax [+V -N] phrase would be initiated. If the order Specifier-Head-Complement is known, then the system automatically creates nodes of these types, on demand for the Specifier and Complements (which may be absent), but obligatorily for the Head (which must be present). So for instance, if the schema Specifier-Head-Complement is being traversed, with X=N, then after finishing with the rules associated with the Specifier packet, the system will automatically create a Head node of type N and make it the current active node. This node will, of course, serve as the attachment point for the actual lexical Head presumably in the input buffer at this point.

Some descriptive details about this process are in order. If the Head is in fact the item in the input buffer that triggers the creation of a phrase, as, say, the name *John* does, then that item is already marked to be the Head of that phrase. This means that if that Head item resides in the first cell of the input buffer that there cannot be any material attached to the *left* of the Head. By similar reasoning, if this Head item is the second or third cell in the buffer, then any material in the buffer to the left of the Head cannot be the Head of the phrase just created; rather, it must be Specifier material (in English), or Complement material (in languages that have Head-Complement order). The acquisition procedure takes note of this condition in its operation.

What of the triggers that Marcus called "leading edges" -- e.g., elements such as "the" or "every" that indicate that an NP must be seen, or adverbs such as "hardly" that indicate the presence of a VP? Observe that these items are all minor categories (-Head); they do not themselves form phrases. In brief, they are not operators. Their category identity is parasitic on the Head category of the phrase of which they are a part. How are the features of these items determined? Consider a simple case, such as the sentence, *Bill ate the candy*, and assume that *candy* is known as +N (because it denotes a physical object). In contrast, *the* is known to be a non-operator, since in neither denotes an object nor an action. Consider what happens when the parse reaches the following state:



The cannot trigger a maximal projection, but *candy* does. The resulting X-max (of type [+N -V]) is made the current active node, and, if the order Specifier-Head-Complement is known, the packet Parse-Specifier-N is entered. If there are no previously known grammar rules to attach *the* as the Specifier of the NP, then the acquisition procedure will be entered. *Attach* succeeds, because the features of the X-max, [+N -V] are compatible with the features of *the*. These features of the N-max may be taken to be the defining features for *the*, along with the current packet (Parse-Specifier-N) active at the time the attachment was carried out. *In other words, the has no definition as a "type" at all, except with respect to the context under which it is inserted as the Specifier of an NP.* This definition of the categorization of *the* in this way seems at first very different from the usual feature complex notation, e.g., [+N -V +Det]. From a slightly different point of view it is not so different, however. A feature complex just defines a non-terminal label of some phrase structure tree under the X conventions. But a non-terminal label just represents the state of some automaton (a push-down automaton, in the case of a context-free system). In a sense then, the feature complexes are just "artifacts" representing the states of a machine. One could just as well define the states of the machine and forget the context-free re-writing system, with its features complexes serving as non-terminal labels, altogether. This is what has been done here. We have defined what *the* is in terms of its behavior in some machine (the modified Parsifal parser) rather than via non-terminal feature complexes.

What would the lexical entry for *the* look like? One way to write it down would be just as indicated above, that is, as a local context-frame something like the following (here we have generalized the third buffer cell to the "wild card" * symbol, that matches any ite as the acquisition procedure would, and generalized the second buffer cell position to remove the literal lexical token "candy".)

the:

```
Cyclic node: INFL-max (S)
Active node: N-max
Packet:      Parse-Specifier

Buffer:
     ----------------------------
     | the [-H] |[+N -V +H ] | * |
     ----------------------------
```

As indicated earlier in this Chapter and discussed formally in Chapter 3, these insertion frames can be described as "local transformations," as suggested by Chomsky [1965] and formalized by Joshi and Levy [1977b]. Plainly though, this insertion frame is not quite general enough, since it includes the "irrelevant" context on the stack, the cyclic node INFL-max (S). This pattern will suffice, however.

In sum, there are two triggers for the creation of an Xmax phrase: (1) the presence of a Head item (as a Noun, a Verb, Preposition, or Adjective -- basically, the class of operators); or (2) the presence of an unambiguous "leading edge" of a phrase of a certain type, a -Head item of type X. Note that this triggering involves a kind of top-down prediction: when a phrase of type X is created, the parser has in effect guessed that such a phrase will eventually be found. At first glance, this would seem to be a departure from a strictly bottom-up or data-driven approach. But it is not. As observed by Hammer [1974], every bottom-up parser makes at least one prediction, namely, that it will be able to build an "S". Thus at the beginning of the parse, a bottom-up analyzer can in fact place an "S" on its stack, knowing full well that if it succeeds at all, then it will in fact be able to construct such a phrase. Starting from this observation, Hammer [1974] went on to show that all such "top-down" predictions could be incorporated into a bottom-up recognition procedure, without destroying the order in which phrases where eventually completely built. Specifically, one could predict that certain phrases would be constructed, and place the names of these phrases on an active node stack, while still completing the construction of these phrases according to a strict, right-most derivation order (in reverse). To do this, Hammer suggested that one think of the construction of predicated nodes as a kind of subroutine call, that suspended the parse of the node currently under construction in order to parse the newly hypothesized phrase. But this is precisely the mechanism Marcus [1980] advanced for handling Noun Phrases. Thus we may view the automatic node creation device as an application of Hammer's "state splitting" subroutine approach to the Marcus parser, extending Marcus' analysis of NPs to all Xmax's.[44]

---

44. Of course, initially the system will not know that "the" flags the creation of an N-max phrase. This comes later, after "the" has been coerced to have the feature +N because of its attachment to the NP prompted by the identification of a Noun head.

There is one additional point to be made about just what the type of a new X-max node should be. If the Head category is known, and there are no other constraints imposed by the X̄ schema system (as is the case, for example, when basic sentence order is completely unknown), then it is the category features of the Head that determine what the "type" of the maximal projection is. So for example, an N projects to an N-max phrase, an item marked +[V, +Inflection] projects to a phrase of type [+V +Infl], and so forth. When items in the input buffer are actually attached as the Head of a maximal projection, then a compatibility check is made to ensure that, in fact, one can attach such an item as the Head of the projection. This check works simply by merging the known features of the maximal projection with the features of the lexical item forming the Head. Note that in the usual case this check will be vacuous. However, it will make a difference in just those situations where there is information known about the maximal projection that exceeds that known about the Head. This happens in the case of Sentences and the INFL element, described briefly earlier. As a later example scenario will illustrate, a lexicalized inflectional element, such as the modal *will*, is initially known as simply a [-N] item -- that is, it is known to be *not* an object.[45] Therefore, this element will be projected as an X-max of type [-N]. Hence, *will* is also marked [-Arg]. The Sentence X̄ schema is already known to demand a [-N +V] Head. Thus, the modal will be *compatible* with this unification check, even if it does not possess exactly the same set of features.

In more detail, coercion is implemented using a feature unification approach that exploits information about the Complement structure of a Head item. When an X-max phrase is created while the Complement portion of a phrase is under analysis, then the features of that X-max (as percolated from the Head) are checked for compatibility against the (obligatory) subcategorization demands of the phrase to which it will be attached as part of the Complement, if there are any such demands. For instance, *want* takes either an NP or an S̄ argument as an obligatory complement. Hence its Complement phrase must include an NP operand, adjacent to the V by default. Suppose then that an example like *I want John to leave* is encountered. In this case, the pattern in the buffer at the time the Complement of *want* is being analyzed will be *NP*, *to*, and *leave*. There are basically two choices: either the NP could be attached as the NP object of the verb, or else the NP-INFL-V pattern could be used to trigger the creation of a new S̄ phrase. Ordinarily, the node creation rule has priority, and this will be the right result in this case. But what about an example such as, *I persuaded Bill to leave*, where *persuade* takes both an NP and S̄ arguments, and hence where *Bill* is an Object NP? Here the protocol to create an S̄ will form the wrong structure: [I persuaded [Bill to leave]]. But note that in this case the subcategorization frame of *persuade* may be assumed to have been previously established as [NP S̄], with the order fixed by a positive example. The creation of a S̄ violates this frame, since no NP

---

45. More precisely, modals such as *will* are unlike verbs in that they do not take NP complements. We could use the feature "-Arg" to express this fact. Thus, Verbs will be marked [-N +V +arg], modals, [-N +V -arg]. Jackendoff [1977] used the feature ±Complement to indicate the same thing.

argument has been attached as yet, as indicated by the lack of the right annotation of the VP node. This example shows that automatic node creation must be checked by unification against the subcategorization frame of an Operator (i.e., its Operand structure).

This method requires that we modify the automatic node creation protocol somewhat, in order to incorporate subcategorization information It results in a complex combination of top-down feature coercion and bottom-up data-driven feature specification. Basically, the idea is to use the features of items in the input stream to drive node creation when nothing is known. use subcategorization information to drive node creation when almost everything is known, and to combine features in between. A known subcategorization frame is used to force an X-max phrase to be of a certain type. If the features of the X item forming that phrase are unknown, then they will be forced by that of the subcategorization frame itself. For example, in the *persuade* case, suppose that the NP-S context has been fixed by other examples, such as, *I persuaded John that he was a fool.* Then, when a less clear-cut example is given, this frame can be used in a top-down way. After the argument structure of persuade has been acquired, then upon entering the Complement portion of the analysis of *persuade* in *I persuaded John to go*, the system will require that whatever X-max node be built have the features type +N -V, corresponding to the expected NP. Omitting details, *Bill* will be parsed as such an X-max. When the phrase is completely built - a simple matter in this case -- then it will be dropped into the input buffer. But now, even though S-creation is triggered, it does not pass muster, since it fails the subcategorization check. Hence, we pass to consider grammar rules. Here, the NP is attached to the VP node. Having completed the NP portion of the subcategorization frame, the system will now consider the S. Once again, the system will demand that the features of whatever X-max is built be compatible with those of S. In this case, the X-max projected from *to* will be so compatible. In addition, since it is known that an S is being constructed, the NP-INFL-VP form of an S is now expected.

To take another example, consider again *John wants Bill to go.* Suppose that *want* is known to take either an NP or an S, e.g., *I want very much for Bill to win.* The check for this condition will be simply the feature +Argument -- the element common to both NP's and S's.[46] In effect, the entry for *want* demands either an NP or an S, but this requirement is not stated disjunctively.[47] Now, on entering the analysis of the Complement of *want* in ital(John wants Bill to go), the system will check for the +Arg requirement. Therefore, S-creation succeeds in this case. *Bill* will not be attached as an NP Object, but as the Subject of the embedded clause. Interestingly, as will be discussed below, the example *Bill wants to go* is actually easier to analyze, in the sense that this choice does not arise. In fact, these examples are produced earlier by children -- a result, perhaps, of non-ambiguity in parsing rather than

---

46. This feature serves the same purpose as Stowell's [1981] Case Recipient feature.
47. See Chapter 3 for a formal discussion of why disjunction in general should be eliminated, based upon ease of acquisition.

evidence of a dual subcategorization (as has sometimes been suggested).

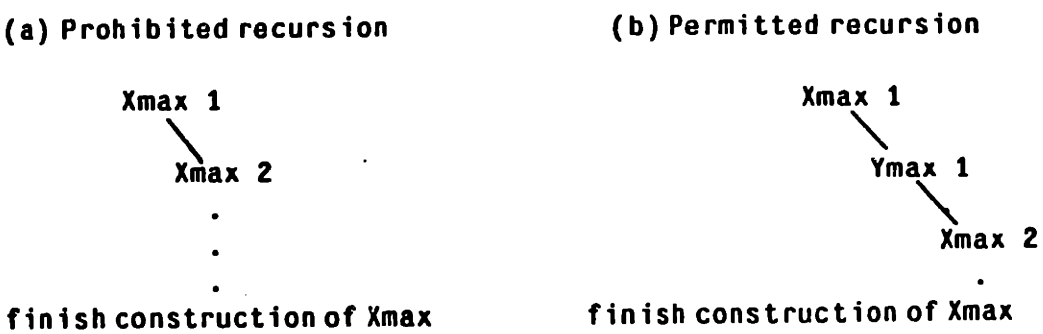
How does node creation fit into the general plan of parsing? Basically, a new X-max node is created whenever a Head item is detected, in a data-driven way. Therefore, this node creation takes precedence over all other operations. Before any grammar rules are tested to see if they will execute, the parser first checks to see if the first item in the buffer can Head a maximal projection of some type. Thus this method will always interrupt parsing to handle an NP, what Marcus called an "attention shift." Again, a special protocol is used to handle the case of S phrases, and is discussed below. Basically, once NP--INFL--VP order has been set, the S schema is used to force the creation of nodes of this type. This "top-down" coercion is important for the category identification of NP's such as *there*, and in the proper analysis of embedded S's with missing Subjects; see Section 6 of this chapter for details.

The focus on Head items as the "core" of parsing is closely related to the parsing strategies proposed by Fodor, Bever and Garrett [1974]. They observed that one way of parsing is to simply look for lexical Heads (Verbs, Nouns) along with their Subcategorization frames; phrases tend to cluster around these items. This suggestion is also compatible with the Projection Principle, which is virtually a restatement of this heuristic. One additional computationally-oriented comment is perhaps in order at this point. There is nothing that would prevent us from attempting to project whatever is in the finite 3-cell buffer *in parallel* -- in fact, this would seem to be a reasonable way to model parallel processing in a basically left-to-right system. When this is done, then the model also bears a striking resemblance to previous A.I.-based parsing systems called "word expert parsing," where each word acts like a miniature finite-state automaton that can inhibit or support its fellow words. (Small [1978]) Without going into an extensive discussion of such systems here, suffice it to say that the approach sketched here is not incompatible with these more heuristically-based approaches; indeed, one way of viewing the buffer and insertion-contexts is as a more careful and linguistically-motivated version of this general approach.

The restriction that the triggering item must flag a <u>new</u> X-bar phrase of a given type deserves additional discussion. Recall that the Instantaneous Description of the parser's state at any given time includes a <u>current</u> <u>active</u> <u>node</u>, the first accessible node on the stack of nodes (constituents) under construction. The restriction of node creation to only "new" Xmax items means that the Xmax phrase to be created cannot be identical to the node currently active. (It could of course be of the same type as some <u>other</u> node on the active node stack.) The intent of this restriction is clear: given the two items "the" and "girl" occupying respectively the first and second cells of the parser's input buffer, we do not want to first create an Nmax phrase via the triggering action of "the", and then, immediately afterwards, create yet another Nmax phrase because the item "boy" is also of the right triggering class. In short, <u>adjacently</u> recursive nodes of the form Xmax <u>directly</u> dominating another Xmax are forbidden. While this constraint works correctly in the simple case above, it seems (at first) plainly at

odds with another fact about language, namely, the existence of "self-embedded" constructions. Noun Phrases can contain Sentential Complements, in turn containing other Noun Phrases, as in *the boy that I like*. Note, however, that the Nmax recursion has been mediated by an intervening S (rather, S-bar) node. By the time the second Nmax needs to be triggered, the currently active node will not be the *current* active node (so blocking entry into a new Nmax schema instantiation). Instead, the interposition of the S will allow the required recursion to occur. In short, self-embedding is permitted in this scheme, but only if some other phrasal node (or lexical material) is interposed between (otherwise directly) recursive categories. Indeed, this arrangement appears to hold quite generally in (English) phrase structure.[48]

This principle has a suggestive computational interpretation, first noted by Chomsky and Miller [1963]. If we imagine entry into an "Xmax" schema as a subroutine call, then each such invocation establishes a "local environment" (in the usual sense of block-structured programming languages). We can picture this by drawing a box around each Xmax node, as in the figure below. By "local" one means that a node can make reference to the environment immediately adjacent to it -- that is, to itself and the routine that called that node. For example, in part (a) of the figure below, the inner-most Xmax, Xmax 2, can refer to material in the Xmax 1 immediately outside of it (including Xmax 1). But in part (b), Xmax 2 can only refer to material in its own environment and in that of the immediately adjacent Ymax 1 environment; it cannot refer to node Xmax 1. Now, observe that in (a) there exists a possibility of confusion in a reference to "Xmax" during the processing of the inner-most node: either Xmax 1 or Xmax 2 is a valid Xmax node. The same opportunity for error is not present in situation (b); here, Xmax 1 is not accessible while processing Xmax 2, and a routine that mentions "Xmax" can only mean "Xmax 2".



This protocol handles the creation of major phrasal categories (Xmax's), but not flanking Specifiers or Complements. Since these constituents are often missing in a given phrase, (that is, the lexical items

48. To take another example, recursion in Noun Phrases often passes through an intervening X-max phrase of type PP.

dominated by these nodes are optional -- sometimes absent, and sometimes subject to the demands of subcategorization frames), it is clear that the creation of nodes dominating Specifier and Complement material should be done on demand. That is, an Xmax schema is not entered *predicting* the existence of Specifier and Complement material -- creating branching slots to be filled on either side of the Head. Rather, these nodes are created only in response to triggering evidence in the input stream.

Finally, it must be stressed that *the actual attachment of lexical material to nodes is carried out by Parsifal grammar rules.* That is, even though the X-bar/packet system directs the creation of nodes and determines whether material may be attached to certain nodes according to base phrase structure rules, actual attachment is carried out by the action of some grammar rule. For example, the following hypothetical grammar rule would build part of an N-max:

```
{RULE Phrase-structure-rule in packet Parse-Noun
IF      1st item in buffer is a Noun [+N -V +M]^0
THEN    attach first item in buffer to current active node as N^0}
```

This rule would have the effect of attaching a Noun to the N-max tree under construction, as desired.

How is it known when a phrase is completed? First of all, the end of a constituent is unambiguously marked by the end of a sentence. For example, the PP, *with red hair* is unambiguously terminated in the sentence, *I kissed the girl with red hair.* Note that the end of sentence marker also terminates the NP (and the VP) in this case. Second, a phrase is unambiguously terminated if it is a single name, such as *Bill.* So for example, in a sentence such as, *I kissed Sally behind the barn*, the PP *behind the barn* cannot be a part of the NP *Sally.* What happens in truly ambiguous cases, e.g., sentences where PPs can be attached either to NPs or to a matrix VP, as in the classic, *I saw the man on the hill with the telescope*? Is this a problem for acquisition?

It is claimed that while these sorts of examples might pose a problem for the thesis that parsing can proceed completely deterministically (in that only a single labelled bracketing is recovered at a time), they do not cause any problem for *acquisition.*[49] The reason is for this is that while there are ambiguous cases where the trailing edge of, e.g., an NP, cannot be pinpointed for certain, there are other cases that are clear-cut. Ambiguities arise out of the interaction of two separately acquired pieces of knowledge. For example, one piece of knowledge is that NPs can have PP complements; this

---

49. One potential confusion should be avoided here: it is well-known that there are parsing methods (e.g., the Earley algorithm [1968]) that build a representation of all possible parses of sentences wrt a given context-free grammar deterministically. This possibility has been suggested by Marcus under a different name ("pseudo attachment") as a way to handle PP phrase ambiguity.

knowledge is triggered by unambiguous positive examples such as, *The boy with red hair is sick.* Here, it is known for certain that the PP *with red hair* ends the NP, assuming that it is already known that VPs cannot appear in the complements of NPs (as must be already known in order to handle simple declaratives successfully). Similarly, an acquisition procedure can acquire the knowledge that VPs can have PP complements, via examples such as, *I saw Bill with the telescope.* No difficulties are encountered *learning* these constructions, but together they form a potent combination: the grammatical system that is induced is ambiguous:

NP-->N (PP)
PP--> P NP
VP--> V NP (PP)

Since this grammar is ambiguous, by a well-known result [Knuth, 1965], it cannot be incorporated into an LR(k) parser, or embedded as is into a deterministic Marcus-type parser. This is not a paradoxical result. There is no reason to expect that the entire system of linguistic knowledge that is acquired must be "processable". To take a more familiar case, consider the existence of center-embedded sentences. As Chomsky and Miller [1963] observe, such sentences cannot be easily processed, if at all, by people; yet they are perfectly well a part of the system of knowledge that is acquired.

There is a slightly different way to look at this difference between acquisition and parsing that shows just why acquisition can be easier than parsing, with the result that a broader class of languages can be easily acquired than can be easily parsed. The acquisition procedure has been designed so that if the parse fails even after the acquisition procedure has been invoked, then analysis of that sentence is abandoned. If this happens, all is not lost; the system will be able to try again, when (and if) the sentence is encountered again. If the sentence is simple enough, then it is reasonable to suppose that it, or a similar construction, will appear again. (Note that this is the same assumption made by Wexler and Culicover [1980] in their mathematical model of the acquisition of a transformational grammar.)

Given this method, the acquisition procedure is non-deterministic *even though any individual sentence analysis is deterministic.* Assuming that simple sentences appear often enough, the acquisition procedure keeps trying its hand at the same sentence until one analysis succeeds. This corresponds to the definition of "success" for a non-deterministic computation: a non-deterministic computation succeeds just in case any one of its computation paths succeeds. Why doesn't the acquisition procedure make the same (incorrect) choices each time it enounters the same sentence? There is one possible reason for a different sequence of choices: new rules have been added since the time the sentence was last encountered. That is, the rule database may have been altered. Of course, for this approach to work there must be some way to detect when matters have gone hopelessly astray. Two criteria have been used: (i) total failure of the acquisition procedure; or (ii) violation of the supplied predication representation.

Note that it is still the case that any single analysis is deterministic. At each step of any given parse, at most one choice can be made. The difference is that different possibilities can be explored *over the course of several sentences.* It is this possibility of "global" non-determinism that broadens the power of the acquisition procedure.

2.4.4.3 Starting a parse -- the Sentence node

The original Parsifal machine moved items from the input stream into the buffer on demand -- that is, only when a grammar rule triggering pattern called for a pattern match against the first, second, or third buffer cells. For example, suppose some of the grammar rules in a currently active packet called for pattern matches against the first buffer cell, and other rules demanded matches against the first and second buffer cells. To ajudicate such a match, at least the first two items in the input stream must be pulled into the first two slots of the buffer.

However, initially Lparsifal has no grammar rules. Therefore, there is nothing to demand that any input tokens to be read into the buffer at all. Without some way of automatically reading in at least one token, the parser would simply stop dead. The procedure must also create some initial current active node. Otherwise, there will be no node available as an attachment point for items in the buffer. In the original Parsifal system, this start-up difficulty was handled by a special initializing rule performed exactly these actions: (1) create an S(entence) node to prod the parse into motion; (2) set up that S node as the initial current active node; and (3) activate two grammar rule packets (a general clause level packet and a packet to parse the constituents at the beginning of sentences):

```
{RULE INITIAL-RULE IN PACKET NOWHERE
IF      1st item in buffer is anything
THEN    Create a new S node.
        !(setq s c).
        Activate CPOOL, SS-START.}
```

This solution seems ad hoc. For one thing, having a special rule to create S-nodes just for the initial setup of a parse is unmotivated (and unnecessary) if in all other cases "ordinary" grammar rules can be used to spark the appropriate creation of Sentenial phrases. But it is certainly true that Sentential nodes must be created all the time in a data-driven fashion; as an example, take any embedded sentence, such as *Sue thought that I was kissing Mitch.* Here, the embedded S *I was kissing Mitch* initiates the creation of an S node -- yet the parser needs no special rule to do the trick. Rather, a data-driven grammar rule detects the tell-tale "leading edge" of a Sentential form -- a Noun Phrase followed by a Verb -- and so creates a new S node. Listed below are some of Marcus' Parsifal rules

that dealt with the creation of S nodes in a data-driven fashion:[50]

{RULE WH-RELATIVE-CLAUSE IN NP-COMPLETE
[relative-pronoun-np] [t] -->
Label the current active node modified.
Attach a new S node labelled Secondary, Relative to the current active node.
Activate CPOOL, PARSE-SUBJECT, WH-POOL.

{RULE THAT-S-START IN CPOOL
[comp, *that] [np] [verb] -->
Label a new S node Secondary, Complement-sentence, That-sentence
Attach 1st item in the buffer to the current active node as Complementizer.
Attach 2nd item in the buffer to the current active node as Noun Phrase.
Activate CPOOL, PARSE-AUXILIARY.}

{RULE INF-S-START IN CPOOL
[*for] [np] [*to] -->
Create a new S node labelled Secondary, Complement-sentence, Infinitive-sentence.
Attach 1st item in the buffer to the current active node as complementizer.
Attach 2nd item in the buffer to the current active node as Noun Phrase.
Activate CPOOL, PARSE-AUXILIARY.}

{RULE INF-S-START1 IN INFINITIVE-COMPLEMENT
[np] [*to,auxverb] [tenseless V] -->
Create a new S node labelled Secondary, Complement-sentence, Infinitive-sentence.
Attach 1st item in the buffer to the current active node as Noun Phrase.
Activate CPOOL, PARSE-AUXILIARY.}

With all these ordinary grammar rules to create S nodes, why should the very first S node created have any different status? Note that Parsifal creates other category nodes -- notably Noun Phrase nodes -- in exactly the same data driven fashion as the S node creation rules above. If one adopts the data-driven creation of nodes as the normal way of life for Lparsifal, then no special parse-initializing rule is needed at all. Instead, let us regard S-formation as an extension of the "Xmax creation" protocol discussed earlier: given an appropriate "trigger" in the buffer, the parser should *automatically* create a new S node and make it the current active node, entering (and activating) the X-bar schema

---

50. Certain liberties have been taken here with Marcus' English-like notation for expository purposes. The priority system has been omitted, certain abbreviatory notations for packet names, pattern-matches, and labelling features have been expanded, and so forth.

packets of an appropriate type. The question now becomes, What sort of Xmax is an S node, and what (if any) triggers should be used to flag it?

Recall that S has been assumed to be assembled from the elements, {NP, INFL, VP}, with the INFL, VP pair being conflated as INFL/VP initially. INFL has been assumed to be the Head of S (INFL/VP if node conflation is assumed). Given this analysis, we could use the same node creating protocol presented earlier for other types of phrases to create S phrases. First, if the Head of S -- INFL -- is unambiguously recognized, then an S node should be created and placed on the active node stack. Second, if the "leading edge" of S, the Specifier of INFL in English, is unambiguously recognized, then an S should be created. (This does not mean that it is possible to recognize the leading edge of an S, of course.)[51]

Like other node creation rules, this protocol takes priority over the execution of grammar rules. So for example, suppose that the trigger for S's is an NP followed by some +V lexical item bearing an Inflection, or simply NP followed by a Verb. As usual, this creation is subject to feature compatibility checks. Finally, we add the special requirement for S's that the schema NP--INFL--VP (N-max--INFL-max--V-max) is to be followed literally, once the order Specifier-Head-Complement is fixed, and once NP Specifiers have been established by positive evidence. That is, if an "S" is created, then the system next expects the Specifier to be an N-max, and will use this requirement as a feature compatibility check before moving on to the analysis of the Head portion of the S.

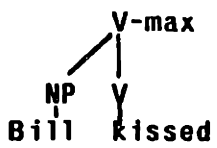
In practice, it may not be so easy to detect an INFL element. Consider what happens if the inflectional element is morphologically attached to the verb, as in *Bill kissed Sue.* In this case, *Bill* is projected as an N-max, i.e., an NP. What about *kissed*? Suppose that it is known as a verb (an action) i.e., it is marked [+V -N], but that the *tense* feature is not known. Then *kissed* is projected as a [+V -N]-max. Next, the NP just constructed may be attached to this projection, since this construction meets the basic (semantical) constraint that NP arguments are incorporated *into* maximal projections of [+V -N] elements, rather than the other way around. We arrive at the following structure, with INFL and Vmax (VP) nodes conflated:[52]

---

51. We put aside for now discussion of the complementizer system in English, i.e., constructions such as *that*-S, or *for*-S. This is covered later in Section 6 of this chapter.

52. An alternative would be to analyze S as NP (Subject) Predicate, with Predicate as INFL VP.

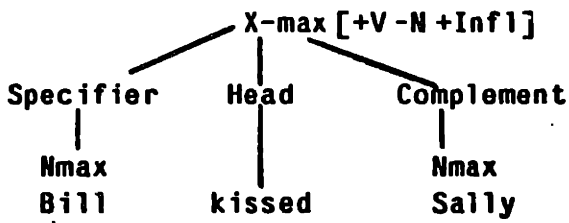
(partial tree)



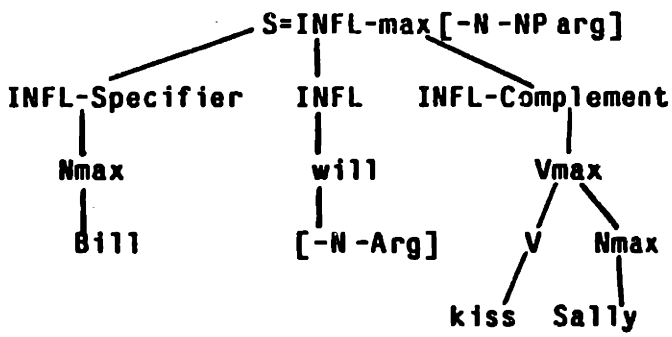
words remaining: Sally

Interestingly enough, there seems to be some evidence of "flat" structures such as these in the language of young children; see Tavakolian [1977].

Now suppose that the tense element is recognized as such, a fact that might as well be notated by a diacritic +Infl. Then by the way we have defined category identity, the INFL and VP non-terminals will remain conflated by examples such as, *John kissed Sally*. The only difference will be that *kissed* will spark the formation of an Xmax node of type [+V -N +Infl]:



How could the correct INFL node be acquired then? Again the process is similar to the formation of new categories in the automata theoretic sense. When an example appears with a separate INFL element that is *not* a verb (i.e., elements that do not absorb NP arguments, like modals) then this lexicalized reflex of INFL is attached as the true Head of INFL, the following verb becomes the Complement of INFL (and is also a VP):



Here, we have used the feature "-Arg" as a gloss for the fact that modals do not take NP complements. Note that this is only a description of the way things are, not an explanation; it does not say *why* it is that *will* does not take NP complements. We could just have well stored the insertion frame for *will* as we did for *the* earlier, that is, without an explicit feature to serve as the proxy for the machine state.

Finally, observe that the INFL state is now distinguished *in terms of parsing* from the maximal projection of V. Before this example is encountered, the V-max and INFL-max nodes are collapsed together. This means that an "ordinary" declarative sentence *without* an explicit Inflectional element no longer fits into the NP-INFL/VP paradigm. The only way that it can be made to fit now is to posit a separate, abstract element -- call it a "tense" or "inflectional" element that is attached to the Verb morphologically but may be split off from it. The example scenarios below describe this process in more detail.[53]

This line of reasoning may well be too strong. As sketched here, INFL is set as the Head of S because in English there is a separate category, the Modals, that is not the same as the category of Verbs. But what about other languages where there is no such independent category? Then there would be no lexical reflex of INFL, and hence no reason to distinguish INFL from V as the Head of S. The INFL projection node would never be split from the projection of V. Turning to intermediate, there could be languages where there is a separate class of auxiliary-verb like elements, and these could prompt the formation of an X-max of the appropriate type to head S. (See Akmajian, Steele, and Wasow [1979]). If this way of inducing the Head of S category is incorrect, then one could simply insist that S's (propositions) consist uniformly of the elements NP INFL VP, with INFL lexically realized or not.[54]

A summary of the feature system and its development is perhaps in order at this point. First, the system starts with just two basic categories, ±N (grounded on the semantic notion of Object) and ±V (grounded on the semantic notion of action).[55] It may be possible to start with just a single category, ±N, and form the category ±V via the process of arborization described by Keil [1979]. If a full sentence is composed of an item that is +N and an item that is -N, and it is also known that a full sentence must consist of a logical Subject plus some "operator" that is -N, then the new operator

---

53. It may be that the separate status of the INFL element can be determined by other linguistic processes. So for example, the use of Auxiliary-inverted questions and *wh*-questions could give rise to the form, Operator-Complement, where Operator= an Auxiliary Verb; and Complement= a Proposition. For example: Will [Sally kiss John]; Can [Sally kiss John]; Why [did Sally kiss John]. (Here we assume an extension to the X system to include an S phrase of the form Operator-Complement.) In fact, something like this appears to occur in child language acquisition. There is a high proportion of auxiliary-inverted forms in mothers' speech to children [see Newport, Gleitman, and Gleitman 1977]. This may establish the auxiliary verbs as a separate category, distinguishing them from main verbs. Note that the form [Kissed Sally John] is not encountered.

54. This is one place where an analysis of S as Subject-Predicate might be advantageous.

55. Perhaps this should be predicate rather than action?

category could be identified as the ± feature. This matter is completely speculative, of course. The feature +V is based on the notion of "action" but it is not exactly coextensive with the intuitive sense of "action." It is closer to the idea of "thematic role assigner."[56] Intuitively, if sentence meanings are interpreted as representations of events, and if events are composed of actors or events linked by actions or thematic role assignments, then +V elements form the links of this quasi-pictorial representation, and +N or S items the nodes.[57]

In any case, besides the two initial categories, one must also have a way to distinguish between elements that can form maximal projections and those that cannot. Nouns, Verbs, Prepositions, Adjectives, and, if we consider INFL as the Head of S, modals, can all form maximal projections, acting as operators. They are marked +Head. Particles, Determiners, and Adverbs cannot form phrases, are not operators, and are marked -Head. How could this be learned? Determiners, Adverbs, and Particles will never appear as either predicates or as arguments in given predication structures, nor will they appear in the proper position of predicates and arguments, once basic Operator-Operand structure has been fixed for the language at hand. Finally, there is the feature *Argument*; +Argument items take NP operands; -Argument items do not.

Before moving on to actual acquisition scenarios, let us summarize the major X̄ assumptions that have been made here, particularly those that relate to the operation of the acquisition procedure itself.

1. Initially, X̄ structure is based on a known (given) predicate-argument structure that consists of a Logical Subject and a Predication. An early confluence between semantic and syntactic representations is assumed. The syntactic counterpart of this representation is the unordered set of elements, {NP, Inflection, VP}.

2. Heads of phrases are operators, taking operands. They project to maximal X-phrases of a type corresponding to the category identity of the Head. Non-Heads are never operators, and never appear as predicates or arguments. Non-Heads do not form phraes.

3. The inflectional element is the Head of S. It is a composite category initially, having verb-like characteristics. Later examples split this composite category into Infl and VP units. In addition, the order of {NP, Infl, VP} is set by positive examples, and this information is saved, so that the Specifier of S is known to be the NP, and the Complement, the VP.

4. More generally, non-terminal labels defined as clusters of equivalence machine states, where the machine is the Lparsifal analyzer. Lexical items are stored according to their local context-sensitive

---

56. Following the suggestion of Stowell [1981].
57. The reader should note the similarity between this proposal and the usual case frame theories, e.g., Fillmore [1968].

insertion frames. Classes of lexical items are defined by common insertion contexts. Positive examples are used both to form merged classes ("generalization") and split perhaps previously merged classes ("refinement"). The development of lexical classes is therefore a dynamic one.

5. The order of {Specifier, Head, Complement} is used to automatically create an obligatory Head node, and, in a data-driven fashion, a Specifier or Complement node. In the special case of Sentences, the order is used to automatically create NP, INFL-max, and VP nodes.

6. Feature unification is used to determine if a lexical item can be attached to a phrase under construction. This unification process can also be used "top-down" to fix the features of unknown items (e.g., Determiners).

7. Constituent completeness is initially determined by a syntactic-semantic correspondence.

Let us trace through a specific example of X-bar schema system in action, without being concerned about how the rules that execute were acquired. Assume that {Head, Specifer, Complement} order has been fixed as, Specifier-Head-Complement, and that the INFL and VP nodes have been distinguished. Suppose further that a simple declarative sentence, e.g., *John kissed Sally*, is to be analyzed.

First, the $\bar{X}$ schema is entered with X = N, as triggered by the name *John* in the first cell of the input buffer. This creates an N-max node and makes it the current active node. The current active packet is set to the cluster Specifier-Head-Complement (since this order is assumed known), and the packet associated with Specifiers, Parse-Specifier-N, is activated. We will ignore the details of Noun Phrase parsing here.

After all possible rules in the Noun Phrase schema have had a chance to match and run, a Noun Phrase will assumed to have been successfully built. The active node stack will be popped, dropping the completed maximal projection of N into the buffer. If possible, a test for this completed phrase can be made by comparing against the supplied predicate-argument structure. The buffer now contains the tokens N-max (NP) and the [+V -N +Infl] marked item *kissed*. This second item sparks the formation of a maximal projection ot type [+V -N +Infl] i.e., an INFL-max or S phrase. At this point, if one is dealing with a "mature" parser, a grammar rule in the packet associated with the Specifier of INFL will fire and attach the Noun Phrase in the buffer to the X-max (the S), as desired. Then the pointer for the X-max system will be stepped to the next part of the base rule, activating the Head of INFL packet. Note that this occurs because the Specifier of INFL is known to take only only constituent, namely, NP. (This rule is peculiar to the S phrase.) The packet pointer is advanced to the Parse-Head position. The current item in the first cell in the buffer is *kissed*, by assumption partitioned as, [kiss +V -N][ed +V +Infl -N]. A *switch* moves the [ed] element into proper position

for attachment as the Head of the S, the analogue of affix hopping. The packet pointer is advanced to parse the Complement of the S, an element that must be an X-max. The first item in the buffer, *kiss*, triggers the formation of an X-max, of type [+V -N] The V-max node becomes the current active node, and the packet schema with X=[+V -N] is entered. With the Verb Phrase packet activated, the construction process repeats itself. If a Verb Phrase truly follows the Noun Phrase, it will ultimately be completely built, and finally attached to the "S". As usual, the pointer to the S schema will be incremented, but this time, the packet-pointer is advanced past the end of the S schema. The completed S (INFL-max) is dropped into the buffer. If the sentence is also at an end, well and good; the machine enters the accepting configuration, with an S as the first cell in the buffer and the end-of-sentence marker # as the second element, with the active node stack empty.[58]

2.4.5 Base Rule Acquisition Scenarios

Given this implementation, the following scenarios illustrate how Lparsifal can set each of the X-bar parameters by attempting to parse just simple positive example sentences. The initial examples will be somewhat laborious to work through, because so little information is known. As a result, the initial examples also rely more on the assumed given predicate-argument representation. We will begin with the acquisition of rules for the basic NP-INFL-VP order of English sentences. Following this example, we shall spend some time showing what happens if the initial examples given are less than ideal, because of ambiguity in word class identity. What will be demonstrated is that when there are too many uncertainties then acquisition cannot proceed. If word classes are uncertain *and* new grammar rules must be acquired, then the incremental character of the acquisition procedure has the effect of stopping the system from making an unsure inference. If, however, basic NP-VP order is established (by clear examples where word class identity is on safe ground), then new lexical classes can be formed. Following this aside, Section 2.4 concludes with a second example presented to illustrate how a separate INFL node can be split off from a conflated INFL/VP node. Section 2.5 continues with a detailed set of acquisition scenarios for the entire English Auxiliary (Inflectional) system. Section 2.6 concludes the chapter with an analysis of the acquisition of non-base rules.

Example sentence 1: *John kissed Sally.*
Parameter: Left or Right attachment of Specifier (Complement) to Head.

Base rules assumed known: none.
(Noun Phrase building rules are assumed known here for purposes of exposition.)

---

58. If it is not, the situation is more complex; the S just built may be part of an embedded sentence. This particular outcome will not be covered here.
See Chapter 5 for a formalization of the parser, including a precise definition of accepting and non-accepting states.

(Recall also that the acquisition procedure can classify basic word tokens as *nouns*, *verbs*, both, or *other*.)

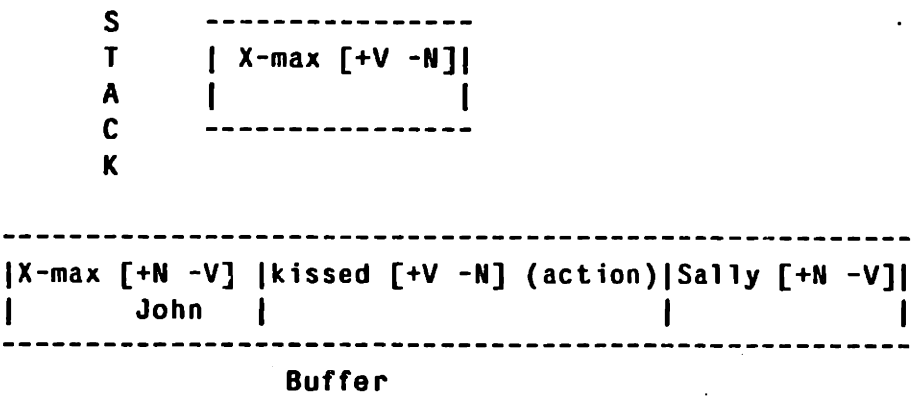
Predication known: Predicate: kiss Sally; Logical Subject: John.
Categorization known: John, Sally [+N -V] kiss [+V -N]

Base rules to acquire:

(1) Specifier Noun Phrase attachment to the left of the INFL/VP.

(2) Complement Noun Phrase attachment to the right of the INFL/VP.

With the first token of the input stream (*John*) entering left-most cell of the input buffer, Lparsifal begins its work. No order is known to the base rule schema {Specifier, Head, Complement}, so no packet is activated. *John* is assumed known to be a proper name (is marked [+N -V]) and so is an unambiguous flag for the start of a Noun Phrase. Therefore, a maximal projection of type [+N -V] is formed, and this node is placed on the active node stack as the current active node. At this point, Lparsifal builds such a phrase, returning if successful a completed N-max node (John) to the first cell of the input buffer. The details of the Noun Phrase parse will not be covered here.[59] Next, the automatic node creation protocol projects the [+V -N] item in the first cell of the buffer as an X-max of type [+V -N]. The parse now looks like this:



59. The only real difficulty in constructing the Noun Phrase comes in knowing that the second token *kissed* (a [+V -N] itme) is not part of the N-max phrase. This would mean that *John* is an operator, *kissed* an operand of this operator. But this is an invalid predication, according to what the system is assumed to know about the "meaning" of the sentence. In effect, since *John* is assumed to be known as a complete semantical unit. Therefore, it cannot have any attached operands. As a result, attachment of *kissed* as part of the complement of the NP fails; *switch* and *drop trace* also fail, because no other grammar rules are known that can fire after either of these actions. In contrast, the system does know that the NP *Sally* is an operand of the predicate headed by *kissed*

Note that as things stand, the Noun Phrase must be attached to the current active node (as the Subject of the sentence). However, there is no known base or grammar rules to do the job; Parsifal must acquire a new rule. As usual, it first tries the *attach* action. This attempt succeeds, since the first item in the buffer, the NP *John* is by assumption known to be the (logical) Subject of the proposition, hence the NP of the Proposition's {NP, INFL/ VP} structure. Furthermore, this node cannot be the Head of the X-max (its features are incompatible, and it is a full phrase, rather than a single lexical item). Therefore, it must be either a Specifier or Complement. The new rule is saved, and the current active node annotated with the name of the rule just executed:

```
Supply a name for rule being created in packet
Unknown

Pattern of rule is:
  Cyclic node: Nil
  Active node: X-max [+V -N]
  Buffer:      [ X-max +N -V][ kissed +V -N][ Sally +N -V]

Action of rule is: attach


>NP_Subject_attach
```

The parser state after the new rule executes:

```
S      ---------------
T      |X-max [+V -N]| current active node
A      |             | RULE NP_Subject_attach
C      |   N-max     |
K      ---------------

-------------------------------------------------
| kissed [+V -N] (action) | Sally [+N -V]|   #  |
-------------------------------------------------
                  Buffer
```

With the NP attached, the verb and remaining lexical token *Sally* remain unanalyzed. What happens next? Since no grammar rules are known, the parse stops at this point, and the acquisition procedure attempts to construct a new grammar rule. It first attempts *attach*. Can *attach* work? Yes, because the first item currently in the buffer, *kiss*, is marked [+V -N], hence can be the Head of a [+V -N] projection. The unification check succeeds. Importantly, *kissed* is marked with the feature +H(ead),

for later reference:

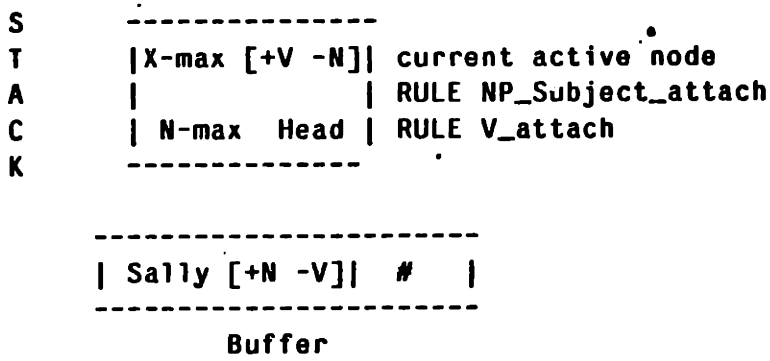
```
Supply a name for rule being created in packet
Parse-Head-X [+V -N]

Pattern of rule is:
   Cyclic node: Nil
   Active node: X-max [+V -N] +RULE NP_Subject_attach
   Buffer:       [ kissed +V -N][ Sally +N -V][ # ]

Action of rule is: attach


>V_attach
```

The parser state after the new rule executes:



This new rule is executed, leaving the token *Sally* as the first element in the buffer. (As always, the active node, the X-max [+V -N], is annotated with the name of the rule just executed.) Next, *Sally* is analyzed as an NP: it is known as a name, prompting the creation of a maximal projection of type [+N -V] This node becomes the current active node. The grammmar rule for attaching *Sally* as the Head of this phrase executes, and the N-max is completely built. Since the end-of-sentence indicator is all that is left in the buffer, the N-max is dropped into the buffer. At this point, no further grammar rules execute, so the acquisition procedure is invoked again. *Attach* succeeds, since a [+N -V] item can be attached as an argument to a phrase with a [-N +V] Head. Since *Sally* is known to be part of the predicate, hence an argument supplied to the V *kissed*, this NP must also be the Complement of the V-max. This information fixes the formerly unordered set {Head, Specifier, Complement} to {Head-Complement, Specifier}, also informing the system that the proper packet active at this time is Parse-Complement-X [+V -N] . The first NP attached as the Subject must therefore be the Specifier

of the X-max, since Complements can appear only on one side of the Head. So the order of elements, Specifier-Head-Complement is fixed by this single example.

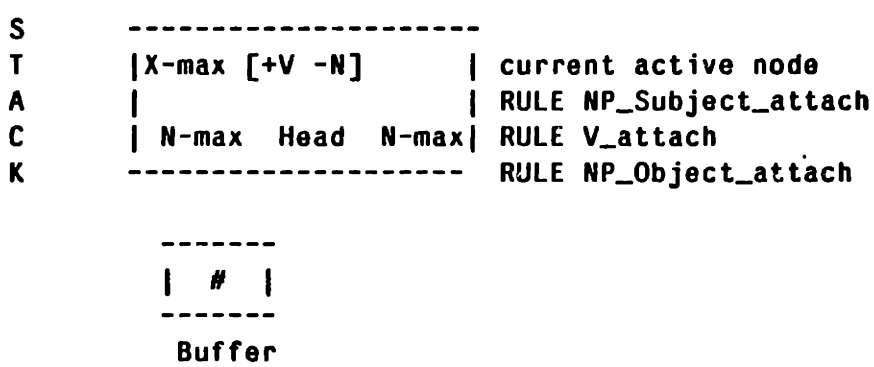
```
Supply a name for rule being created in packet
Parse-Head-X [+V -N]

Pattern of rule is:
   Cyclic node: Nil
   Active node: X-max [+V -N] +RULE NP_Subject_attach RULE V_attach
   Buffer:       [ X-max +N -V][ # ]

Action of rule is: attach
```

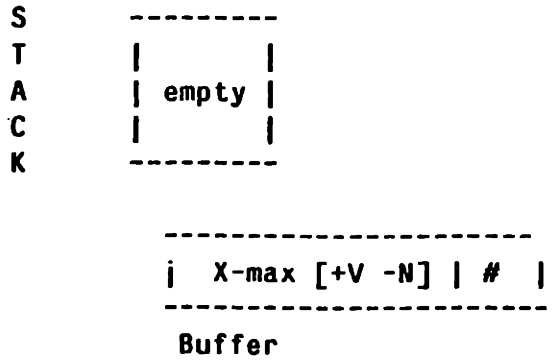
>NP_Object_attach

The parser state after the new rule executes:

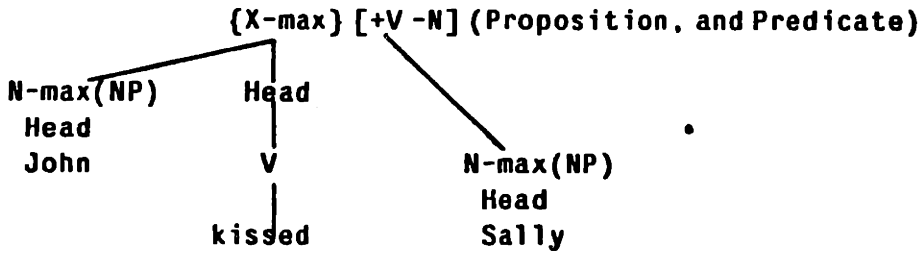


The V-max is now known to be complete (only the end-of-sentence marker remains in the buffer); hence it is dropped into the input buffer. We arrive at an accepting configuration: the stack is empty, and the input buffer holds the two tokens X-max (+V) and #, the end-of-input marker.[60]

60. This is not quite a *mature* accepting configuration, since the INFL element need not be present. As a result, at this stage the device would accept N-V-N structures without inflections, e.g., *I kiss Daddy*. This prediction appears to be roughly correct. One can eliminate this non-uniformity by reducing the X-max back to a "Proposition" and calling this the single accepting configuration.



The output tree built by the parse is as follows:



The known predicate-argument structure can now be assigned to this tree, as indicated. The X-max left in the buffer is assumed to correspond to the Proposition; in this case, it is also a Predicate.

There is no distinct INFL node in the tree. This is a natural result, given the X-bar theory. Maximal projections are based on the category of their Heads. If *kissed* has both the features of being a Verb (an action), and the features of being tensed (possibly an intrinsic property of an action), then its maximal projections will have both INFL and V-like properties, a kind of composite non-terminal. If we regard non-terminal names as just the equivalence classes of the states of a machine as it analyzes sentences, then this is perfectly acceptable; it simply means that not enough information has been received to warrant separating the {INFL, V} categories. A separate "concept" of INFL has not yet been acquired. (As mentioned above, an example where INFL is lexicalized will prompt a category split.)[61]

What other knowledge was acquired via the analysis of this example? Summarizing, the following information was established:

---

61. As noted earlier, the requirement that a lexicalization of INFL must exist in order to prompt a category split may well be too strong.

1. The order Specifier-Head-Complement.

2. The order NP-INFL/VP-NP. (Form: A grammar rule to attach NP's to an Xmax)

3. A grammar rule to attach NP as the Complement of VP's.

4. A grammar rule to attach a *name* as the head of N-max.

5. A grammar rule to attach V as the Head of V-max.

In the preceding example, the categorization of elements as either Nouns or Verbs was clearcut, or else deducible from supplied information about the predicate structure of the sentence. What happens if we supply a sentence with an ambiguity as to its category identity -- say, a lexical item that is used sometimes as a Noun and sometimes as a Verb, like *dog*. There are basically two possibilities that can arise here, depending on the procedure's knowledge base: (1) nothing has yet been learned, not even the Specifier-Head-Complement order of sentences (i.e., the expansion order NP INFL/ VP); (2) basic X order has already been acquired from clear examples, e.g., *John kissed Sally.*

In the first case, the system is still supplied with the proper predicate knowledge. Suppose, for example, that the system is given the sentence, *John dogged Sally* (meaning that John followed Sally around). In addition, the predication representation, L.Subject: John; Predicate: dogged Sally, is assumed known. The NP *John* will be analyzed as before. Now we have the problem of projecting the phrase associated with *dogged.* There are two sub-cases, depending upon whether *dog* is unknown or known as a [+N -V] item; let us cover each in turn.

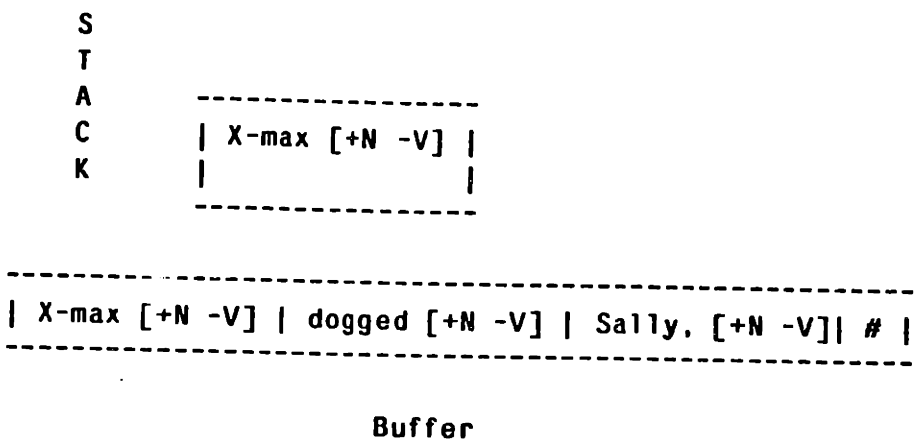
Case 1(i):

First, one might assume that the category identity of *dogged* is completely unknown. Then the maximal projection will simply be of an unknown type. Can the N-max *John* be attached to this maximal projection? There is simply no way to confirm or deny it. Should this be interpreted as success or failure of *attach*? On the one hand, it is success, since the unification check succeeds; since there are no features associated with the X-max, then the compatibility check is vacuous. On the other hand, without predication information, the identity of the maximal projection remains undetermined. The actual result is that a parse tree is built, but with an ambiguous category of type X as the root. (One can imagine a procedure that tries to match this syntactic tree against the known predication information, but this approach has not been adopted here.)

Case 1(ii):

In sub-case (ii), suppose that *dog* is known only as a Noun (an object), (on the basis of previous examples) and that this feature sparks the formation of an N-max phrase. Eventually, *dogged* will be

attached as the Head of this phrase. (Note that this is *not* an incorrect grammar rule, but rather the usual Noun Phrase building rule applied incorrectly because of incorrect categorization information.) At this point, the machine state will look like the following:



Now a problem arises. There is no known grammar rule that executes in this situation, so the acquisition procedure will attempt to construct a new rule. First, it tries *attach*. But the N-max *John* cannot be attached directly to the N-max active node; this would violate the prohibition against direct recursion. It cannot be attached as the Head, since the lexical item that prompted the creation of the current X-max is *dogged*. The actions *switch* and *drop trace* also fail, because no known grammar rules execute after either one is attempted. Therefore, the N-max cannot be attached as either the Head or Specifier, and the acquisition procedure fails complete.[62]

In short, where category information is unknown or incorrect, and no "top-down" information exists on which to base a choice, then the acquisition procedure fails.

Case 2:

In the second case, the expansion order NP-INFL/VP is assumed to be known. Therefore, the packet schema (Specifier-Head-Complement) is attached to the INFL-max (S) node. The parse of the NP *John* proceeds as before. The pointer to the packet structure is advanced from the NP to the INFL/VP component. The active node stack is empty.

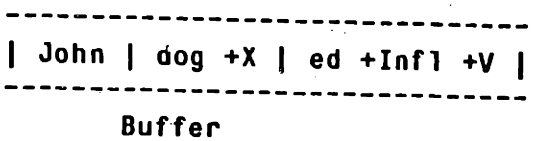
Next, *dogged* is projected to form an X-max, as before. This time, however, there is a difference: since

62. As emphasized earlier, this does not mean that this sentence is uninterpreted. An interesting design alternative here would be to eliminate *all* the grammar rules that were made during the analysis of this particular sentence – that is, to save grammar rules only if the entire sentence is successfully analyzed.

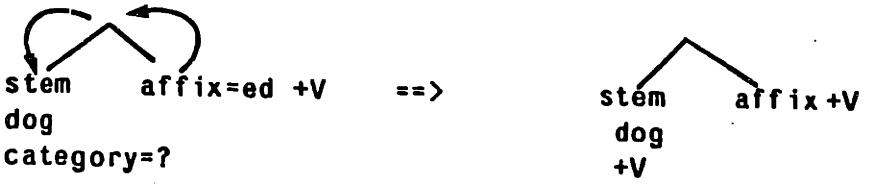
the X schema is known, and the features of the X-max just projected are unknown, a merger of the features results in the X-max being set to the value X=[+V -N]. In effect, *dogged* has been forced to be a V in virtue of known context. The remainder of the parse will proceed just as if the token had been a known V rather than an unknown item.

There is an additional complication to this story, depending upon the system's knowledge of inflectional affixes. Again there are two sub-cases to consider, one where the *tense* affix to *dog* is recognized, and one where it is not.

Case 2(i): Tense affix known.
By assumption, since the tense affix is known as such, the input string is divided into the following tokens:[63]



This is a strong assumption; it posits that there is a "state" identifiable as *tense* and lexicalized as "ed", even though this token has been absorbed phonologically onto the verb stem. Why is it also marked +V? Here, we appeal to a proposal of R. Lieber's [1979], namely, that lexical stems are never marked Noun or Verb. Rather, it is context of the affix that establishes the +V status of the stem.

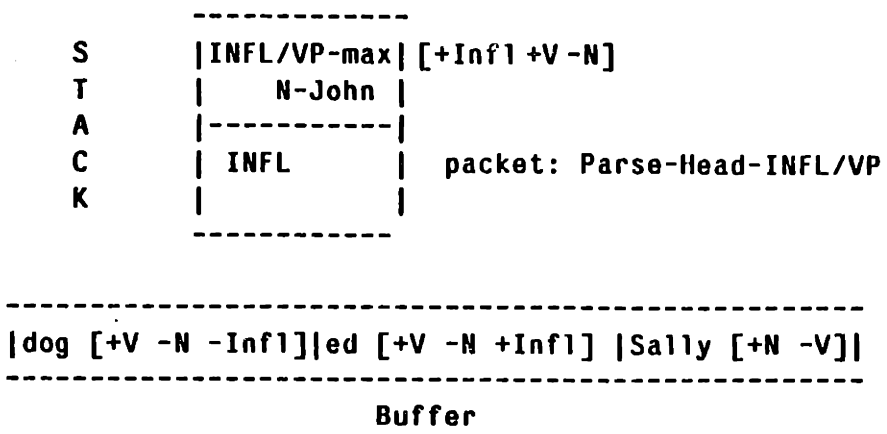


Hence, in this case *dogged* can actually be inferred as a +V item; presumably, this is the productive verbalizing process that is used even by adults.

In any case, given this segmentation, by the time *dogged* is reached in the parse, the state of the machine will look like the following. (Here we will use the +Infl feature as a gloss for the insertion context associated with inflectional elements like Modals -- the alternative feature used earlier was ±Argument.)

---

63. How can this assumption be justified?



No known grammar rule executes. *Attach* fails, because the +V item *dog* fails the feature compatibility check. However, a *switch* succeeds, because the pattern after the switch is the same as in an example such as, *John will kiss Sally*. (Note therefore that this particular example will be successfully analyzed only after the INFL element has been split off from the VP node by an example with a lexical INFL element, so that the "state" represented by the +Infl diacritic is known.) Additional examples of the induction of the INFL system are given below.)

Case 2(ii): Tense affix unknown.
Suppose, however, that the *ed* affix system is simply unknown, along with the category identity of *dog*. If the NP-INFL/VP order is known, then *dogged* is forced to have the features of the INFL/VP node, as described above.

Having discussed the effect of ambiguous or unknown category identity, let us turn to the concluding example of this section. This sentence demonstrates how a separate INFL node can be formed.

Example sentence 2: *Sally will kiss John*

Base Rules assumed known: all those acquired in Example sentence 1.
Predicate structure assumed: Predicate: kiss John; Logical Subject: Sally.
Lexical categorizations known: Sally, John [+N -V +H], kiss [+V -N +H], will [-N] (Names and *kiss* were marked +H by the previous example.)

Rule to acquire: Attachment of *will* to the INFL phrase; expansion NP INFL VP.

A key difference with analysis of this second sentence is that the basic constituent order for sentences, NP-INFL/VP has been established, with the Specifier NP having the features [+N -V] and the INFL/VP component having the features [+V -N]. That is, the system now assumes that the order

NP-INFL/VP-NP will be observed. It therefore now uses this order as part of its compatibility check to see if grammar rule attachments are permitted or not.

Beginning the parse, *Sally* triggers the creation of an N-max. This new X-max is checked with the compatibility condition established by the order NP-INFL/VP, and, of course, it passes this check. Next, parsing of the N-max is initiated. Omitting some details, *Sally* will be attached as the Head of this N-max. Given that Head-Complement order is now established, the packet pointer will be stepped along to the Complement portion of the N-max analysis. Now, no known rules can execute, and the system attempts to acquire a new rule. The current state of the machine is as follows:

```
S       -------
T       |X-max| [+N -V] packet: Parse-complement-N
A       | Head|-Sally
C       -------
K

        ------------------------------------------------
        | will [-N] | kiss [+V -N +H] | John [+N -V] |
        ------------------------------------------------
                Buffer
```

First, an *attach* is attempted. But this fails the attachment unification check.

Likewise, *switch* and *drop trace* fail, because no known rules match after these actions take place. What happens next? Under the altered definition of acquisition failure, the procedure checks to see whether the X-max just built is complete; it is in this case, it was assumed that *Sally* was known to be a complete semantical unit. Therefore, the X-max is simply dropped into the buffer:

```
S       -------
T       |empty|
A       |     |
C       -------
K

        ------------------------------------------------
        | X-max [+N -V] | will [-N] | kiss [+V -N +H] |
        ------------------------------------------------
                Buffer
```

The system advances its pointer for the NP-INFL/VP check to the next element, INFL/VP. Next, *will* prompts the creation of an X-max phrase of type [-N], making it the current active node. The Specifier packet for this X-max is entered. The current machine state looks like this:

```
S       -------
T       |X-max| [-N] packet: parse-Specifier
A       |     |
C       -------
K

    ------------------------------------------------
    | X-max [+N -V] | will [-N] | kiss [+V -N +H] |
    ------------------------------------------------
              Buffer
```

Next, a compatibility check with the known rule NP-INFL/VP is made, via unification. Observe that the check succeeds, since the X-max [-N] is compatible with the features currently defining the category INFL/VP. Therefore, the current state of the system is as follows:

```
S       -------
T       |X-max| [-N +V] packet: parse-Specifier-X [-N +V]
A       |     |
C       -------
K

    ------------------------------------------------
    | X-max [+N -V] | will [-N] | kiss [+V -N +H] |
    ------------------------------------------------
              Buffer
```

Now the machine state is almost exactly the same as it is for the parse or an ordinary declarative, e.g., *John kissed Sally*. A new grammar rule is constructed to *attach* the first item in the buffer to the current active node.[64] The N-max is attached, leaving *will* as the first item in the buffer. Since the N-max has been completed, the packet pointer for the NP-INFL/VP schema is advanced to the INFL/VP portion, its Head. No known grammar rules fire, but *attach* works because *will* is

---

64. Note that this rule will *not* be generalized with the Subject NP attachment rule previously acquired, because there are two buffer items that are different in this case, the second element *will* and the third element *kiss*, as compared to *kissed* and *Sally*, respectively, in the first case.

compatible with the features of the X-max. In addition, *will* is marked +Head, and its lexical entry is updated accordingly. (This is to distinguish *will* from, e.g., *the*).

With a token attached as the Head of the X-max, the Complement portion of the phrase must now be analyzed. At this point, the [+V -N] item in the buffer, *kiss*, sparks the formation of a new X-max phrase of this type. It is made the current active node (pushing the still active node, the X-max corresponding to *will*, down one in the active node stack. The Specifier packet associated with this X-max is activated, but *kiss* can't be attached as a Specifier, since it had just been assumed to be the Head of this phrase. The Head packet for this phrase is entered, and *kiss* is attached successfully, in the process creating a new grammar rule. Finally, *John* is analyzed as an NP, as before, and placed as complete in the buffer. A new grammar rule is formed to attach the NP to the VP X-max, a variant of the first Object attachment rule.[65] The X-max of type [+V -N] is completed (there is no material in the input stream), and so is dropped into the buffer. Now the current active node is the original X-max of type [-N], and the packet Parse-Complement. The X-max in the buffer is compatible, as a complete constituent of type [+V -N], to be a complement, and so a new grammar rule with the *attach* action succeeds.[66] Since the input stream is still empty, the sole X-max now on the active node stack is dropped into the buffer, ending the parse.

This completes the detailed discussion of the acquisition of some elementary rules for base phrase structure order. We now turn to more complex cases, covering the traces of the acquisition procedure's activity less completely.

2.5 The English Inflection (Auxiliary Verb) System

To show in more detail how rule generalization and rule splitting works, let us consider more fully the acquisition of what are ordinarily called "auxiliary verb" rules. Marcus' grammar for this system is listed below. This list is followed by example scenarios showing how each of these rules is acquired. A formal analysis of this induction is given in Chapter 3, including a complexity analysis in terms of the number of data samples required to induce the grammar for this portion of English.

Marcus' terminology deserves some explanation. The label "c" denotes the current active node, the node to which material from the buffer is attached. Statements of the form "Activate build-aux, cpool," are part of the packet system, eliminated in Lparsifal. "Cpool" is the name of the packet that holds rules that build clauses. The form "=<features>" in a pattern means a predicate test for the

---

65. These two rules will not be merged at this point because the left-hand machine context is different in these two cases.
66. One might want to add the feature +arg to a completed VP to flag predicate completion. Then a proposition would take a +arg constituent.

existence of <features>; the form "=*to" means that the literal string "to" must be present. "1st", "2nd" etc. denote the item in the 1st, 2nd, buffer cells, respectively. Finally, the symbol "t" in a grammar rule pattern is a "wild card," that matches any buffer features. The remaining notation should be self-explanatory.

{RULE STARTAUX IN PARSE-AUX
[=verb] -->
Create a new aux node.
Activate build-aux, cpool.}

{RULE TO-INFINITIVE IN PARSE-AUX
[=*to, auxverb] [=tnsless] -->
Label a new aux node inf.
Attach 1st to c as to.
Activate build-aux, cpool.}

{RULE AUX-ATTACH IN PARSE-AUX
[=aux] -->
Attach 1st to c as aux.
Activate parse-vp. Deactivate parse-aux.}

{RULE PERFECTIVE IN BUILD-AUX
[=*have] [=en] --> Attach 1st to c as perf. Label c perf.}

{RULE PROGRESSIVE IN BUILD-AUX
[=*be] [=ing] --> Attach 1st to c as prog. Label c prog.}

{RULE PASSIVE-AUX IN BUILD-AUX
[=*be] [=en] --> Attach 1st to c as passive. Label 2nd passive.}

{RULE MODAL IN BUILD-AUX
[=modal] [=tnsless] --> Attach 1st to c as modal. Label c modal.}

{RULE FUTURE IN BUILD-AUX
[=*will] [=tnsless] --> Attach 1st to c as will. Label c future.}

{RULE DO-SUPPORT IN BUILD-AUX
[=*do] [=tnsless] --> Attach 1st to c as do.}

{RULE AUX-COMPLETE PRIORITY: 15. IN BUILD-AUX
[t] --> Drop c into the buffer.}

Several comments are in order here. First, the English auxiliary system (or INFL system) is, by all accounts, a simple finite-state system; there is no recursion. As we will see, this fact is reflected in the analysis below, in that rules will operate simply on individual tokens, composed via concatenation.

Given that one can exhibit a finite state automaton for this system, how many states does it take? If we include a start state, a non-accepting state, an accepting state, and a state for the main verb and the trailing NP Object, then Chapter 3 shows that 15 states are minimally required. As Chapter 3 also demonstrates, one can convert this finite state automaton to the following "insertion context" format.

```
Start, Accepting, Non-Accepting states= 3

NP:          [a book]

V-tense:     [____NP]
V+tense:     [____NP]

do:          [____Vmainstem]
             (required to block "I do have taken...")

Vmainstem:   [____NP]

be:          [____Ving]
have:        [____Ven]
modal:       [____V-tense]

passive be

been:        [____Ven-passive]
was:         [____Ving-passive]
was:         [____Ven-passsive]
being:       [____Ven-passive]
```

Total: 15 "states" (insertion contexts).

Turning now to the Marcus sub-grammar for AUX, we note that it is closely related to the insertion context system just described. The `Start-aux` rule corresponds to the start state; the `Complete-aux` rule, to an accepting state. (This is not quite accurate: we should really add the activation of the verb packet as another state, and a trailing NP object as yet a third extra state.) These two rules are replaced by the automatic node creation and completion protocols, and so need not be acquired at all. The packet system is eliminated from grammar rules. The `Aux-attach` rule is easily acquired, as we will see below.

The remainder of the AUX or INFL system has to do with proper insertion contexts for lexical items. There are eight of these, not counting main verb and NP insertion frames. Note that the Marcus

sub-grammar includes a rule for *to* infinitives that is not part of root environments, so we shall not consider that rule further here. Furthermore, the rules for *future* and other *modals* are represented by a single rule in the usual analysis. This leaves us with the following rules in the Marcus sub-grammar for the Auxiliary:

```
1. Do-insertion          2. Modals
3. Passive be            4. have Perfective
5. be Progressive
```

This system of grammar rules directly reflects the grammar for AUX proposed in *Syntactic Structures*, with the additional of a state for passive *be*:

Aux--> Tense (Modal) (have +ing)(be +en)

Since all grammar rules are data-driven, and since we are parsing rather than generating sentences, the optionality of the non-terminals in the *Syntactic Structures* phrase structure rule is accounted for: the system attaches *have* or *be* just in case an element of that type is present; otherwise, that particular option is not selected.

There are some apparent problems with the Marcus analysis, however. While it is perfectly adequate to handle all the occurring grammatical AUX forms, it also accepts some ungrammatical forms -- it over-generates. This could have been expected, since the number of states in the sub-grammar is smaller than that indicated by the minimal finite-state automaton for the Aux system. For example, consider the *do*-insertion rule:

{RULE DO-SUPPORT IN BUILD-AUX
[=*do] [=tnsless] --> Attach 1st to c as do.}

All that this rule calls for is a tenseless verb immediately following *do*. A system containing only this version of *do* insertion will therefore also accept sentences such as, *I did be taking a book.*

Second, there is a problem in distinguishing between the progressive and passive forms of verbs. For instance, Marcus' Aux rules will mark the sentence, *I could have been taking a book* as passive, the same as *I could have been being given a book.* This defect is more apparent than real, however. Actually, the sentence is marked simply as a *potential* passive. When the main verb is finally examined for an "ed" ending this ambiguity is resolved. Thus, this extra state is actually placed in another part of Marcus' grammar. In any case, one must add additional states to distinguish between the *ing* and *ed* verb forms somewhere in the grammar, either in the Aux system itself, or later on. After adjusting for these two problems one can see that the Aux grammar presented by Marcus and the insertion frame

grammar are just the same.

The acquisition procedure can acquire this entire system of rules. A formal analysis is given in Chapter 3; the scenarios below illustrate the actual acquisition process.

In these scenarios, we will assume that NPs are successfully analyzed by already-known rules, and, as usual, that verbs can be identified as such. Finally, we assumed that the *tense* feature is recognizable via morphological endings, as the +Infl feature.

Target rule 1: Modal
Marcus version:

{RULE MODAL IN BUILD-AUX
[=modal] [=tnsless] --> Attach 1st to c as modal. Label c modal.}

Target version of rule:

cyclic node: Nil
active node: INFL-max (X-max of type [+V -N -arg])
packet: Parse-Head-X [+V -N]

buffer: [-N -Arg +H][+V -N +Arg][*]
action: attach

(Recall that "-Arg" is a proxy feature for an insertion context that allows non-argument complements, e.g., +V items, but excludes argument (+N) complements.)

Example 1: Sally will kiss the doll.

Most of this parse will proceed as in Example 2 of the previous section. A key difference appears at the time the acquisition procedure acquires a new rule to attach the modal element *will*. The buffer will differ in precisely one position from the previous case, *Sally will kiss John*, namely, in the presence of *the*. We assume that *the* is known as [+N -V -H].

```
S       --------
T       |X-max  | [-N +V] packet: Parse-Head-X [+V -N]
A       |       |
C       --------
K
```

```
-------------------------------------------------------
| will [-N -Arg +H] | kiss [+V -N +H] | the [+N -V -H] |
-------------------------------------------------------
```

Buffer

Old grammar rule:

```
S       -------
T       |X-max| [-N +V] packet: Parse-Head-X [+V -N]
A       |     |
C       -------
K
```

```
-------------------------------------------------------
| will [-N -Arg +H] | kiss [+V -N +H] | John [+N -V +H] |
-------------------------------------------------------
```

Buffer

Supply a name for the generalized rule being created in packet Parse-Head-X [+V -N]

Pattern of new rule is:
  Cyclic node: Nil
  Active node: X-max [+V -N]
  Buffer:
  [will -N -Arg +H][kiss +V -N +H][+N -V]

Action of rule is: attach

>Attach_modal

By the protocol for generalization, we now have two rules with the same action, the same left-hand context (stack and packet), and buffer patterns differing in just one position -- here, the third buffer cell. Therefore, we generalize these rules by intersecting their features.

A third example, such as, *Sally will go home*, results in a further generalization, removing the specific token *kiss*:

Pattern of new rule is:
Cyclic node: Nil
Active node: X-max [+V -N]
Buffer:
[will -N -Arg +H][+V -N +H][+N -V]

Action of rule is: attach

Finally, an example such as, *Sally will go* reduces the pattern required buffer cell to a wild card, since the end-of-sentence indicator has no features at all. This is exactly the desired result.

Target rule 2: Tensed Verb rule (Simple affix hopping)

Target version of rule:
Cyclic node: Nil
Active node: X-max [+V -N]
Packet: Parse-Head-X [+V -N]
Buffer:
[+V -N][ed +V -N +H T -Arg][*]

Example 1. Sally kissed John.

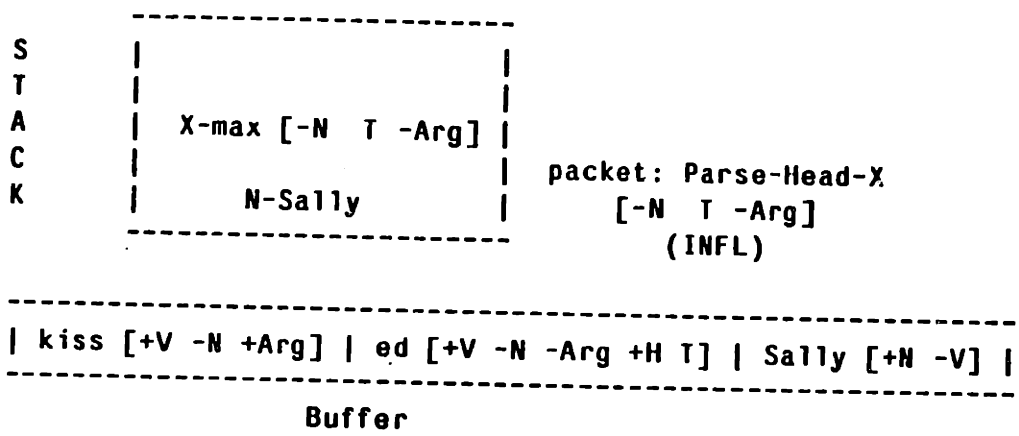
Assume that the INFL/VP node has been split by a previous example, such as, *Sally will kiss John.* This is a natural supposition, as remarked earlier; there is evidence that the separate identity of a "tense" or "Infl" phrase is established via clear cases where there is, e.g., Auxiliary verb inversion. The aim of the affix hopping example is to stimulate the formation of a rule that separates the Verb into its stem+affix components, and then identify the *ed* morphology as the Inflectional element. This is known to be difficult for children, who often drop these markings in speech; note that this fact is expected within the model developed here.[67]

How might an affix be established as a "unit" of inflection? One possibility is to compare the analysis *Sally will kiss John* with that of *Sally kissed John. Will* is a Head item, an operator. Its operand, according to the adjacency constraint, is the Verb to its right, *kiss.* Suppose we say that an operator taking a Verb operand is marked ±T, for *tense.* Then *will* is marked ±T (leaving aside for now the question of whether it is marked + or -.) Now a repetition of the example *Sally will kiss John* will percolate this feature to the X-max (S) node, so that INFL will be required to be marked ±T. But then, *kiss* cannot be attached to the INFL node, since *kiss* is not marked ±T; *kiss* is not followed by a

67. Obviously, there could be sheer processing difficulties, since the identification of *kiss-ed* as two units is more difficult.

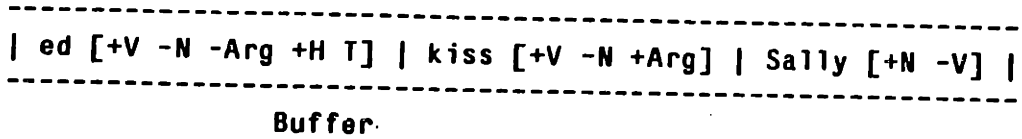
Verb operand. On the other hand, *ed* is a Head, according to the Williams [198x ieber [1980] percolation proposals, and governs the verbal stem to which it is attached. Therefore, we mark it with the feature +H ±T (again remaining neutral as to a + or - marking for T).[68]

Given all of this background, then the system is actually in a position to acquire an affix-hopping rule. The state of the machine at the time the Head of the S (Infl) is parsed is as follows.



The separation of *kissed* into two units simply reflects the assumption that the token is known to be so divisible.

No known rules fire at this point, so the acquisition procedure is entered. *Attach* fails the unification check, because of the incompatibility with the feature *Arg*; recall that a Head must be compatible with its maximal projection. *Switch* leaves the buffer in the following state:



Note that the rule pattern for the attachment of modals triggers on this buffer pattern. (The buffer includes an extra feature, +V, derived from the *ed* ending, but the modal attachment rule does not make reference to this feature at all. Therefore, the predicate demanded by the modal attachment rule

68. Note that this structure violates the operator-operand order for syntactic constituents. I have no explanation for this divergence.

is met by this buffer.) *Switch* has succeeded, and this new grammar rule -- the analogue of affix hopping -- is saved.

Next, the *ed* ending is attached as the Head of the S, the Verb *kiss* prompts the projection of an X-max to form the VP, and the remainder of the parse proceeds normally. Incidentally, note that even if *kiss* is known as both a Noun and a Verb, then the analysis is just as before, with one difference. The core NP-INFL-VP rule is used to throw out the alternative [+N -V] marking for *kiss*; only the [+V -N] cluster survives the feature compatibility check.

Target Rule 3: Perfective *have*
Marcus version:

{RULE PERFECTIVE IN BUILD-AUX
[=*have][=en]-->
Attach 1st to c as perfective. Label c perfective.}

Target acquired rule:

cyclic node: Nil
active node: INFL-max
packet: Parse Head-X

Buffer: [have +V -N -Arg][+V -N][ed +V -N -Arg]

Note: we will ignore the differences in the various inflected forms of *have*, i.e., the differences between *has, have.* We also ignore the ed/en alternation.

Example 1: Sally has kissed John.

There are two separate assumptions to make about what is known for the lexical entry for *have*: (1) *Have* is known as [-N], but nothing more; *have* is known as [-N +V +H +Arg] (a main verb). We shall consider each of these possibilities.

The parse of the NP Subject *Sally* proceeds as before; the NP is built and dropped into the buffer. The NP is then attached as the Specifier of INFL, as usual, and the the packet pointer for INFL advanced to the Head position. What next? If *have* is a main verb, then there is no known grammar rule that matches in the packet associated with the Head of INFL. This is because so far we have rules that demand a buffer pattern like this:

```
-------------------------------------------------
|  -N  T -Arg | kiss [+V -N +Arg] | Sally [+N -V]|
-------------------------------------------------
          Buffer
```

The current buffer does not match this pattern, if *have* is assumed to be a main verb and is marked +Arg. If, in contrast, *have* is just known as a -N item, then the modal attach rule will work. In the sequel then, let us just consider the first possibility, with *have* assumed known as a main verb. The acquisition procedure is entered, and *attach* is attempted. But the *attach* fails, since the INFL Head requires a -Arg item. Evidently, this occurrence of *have* is different from main verb *have*, as indicated by the presence of a +V item immediately adjacent to its right. How can this difference be established? There are several possibilities. One is to exploit the difference between main verb *have* and auxiliary have in examples such as *John has the book* and *John will have taken the book*. Another is to simply set the Arg feature based on immediate local context, instead of retrieving that feature from a lexical entry. An item is +Arg if it is a +V operator in a system with Head-Complement structure, and a +N item appears in surface structure to its right; it is -Arg otherwise. In the case at hand, *have* will be marked -Arg, and therefore can be attached as the Head of INFL. Feature unification marks *have* as $\pm T$ in this context.[69]

Let us adopt the second option for now. Then *have* can be successfully attached as the Head of INFL, via a new grammar rule. Note that since *have* and *will* are <u>not</u> identical, this new rule is not merged with the existing INFl-head attachment rule. As usual, the current active node (INFL) is labelled with the name of the rule just executed.

```
Supply a name for the new rule being created in packet
Parse-Head-X [+V -N T -Arg]

Pattern of new rule is:
   Cyclic node: Nil
   Active node: X-max [+V -N T -Arg]
   Buffer:
    [ have +V -N -Arg][ kiss +V -N +Arg][ en +V -N T -arg ]


> Perfective
```

69. Alternatively, one could simply use the features +V, -N and drop the feature -Arg except as a proxy for the insertion context [___ +V].

Target Rule 4: Modal

Marcus version of rule:

{RULE MODAL IN BUILD-AUX
[=modal][=tenseless]-->
Attach 1st to c as modal. Label c modal.}

Target version of rule:

Cyclic node: Nil
Active node: INFL-max
Packet: Parse-Head-Infl
Buffer:
[-N T -Arg][+V -N +arg][+N -V]

Example 1: Sally could kiss John.

Plainly, this sentence is analyzed just as *Sally will kiss John.* There is one difference. A new *attach* rule is created to handle *could*, since the existing attach rule deals only with *will.* But now we have a situation for rule generalization: a common *attach*, and an identical left- and right-hand side context (save for the two items attached, *will* or *could*). The new rule in effect creates a class of lexical items that call for a certain left- and right-hand insertion context; this class is commonly called the class of "modals."

```
Supply a name for the generalized rule being created in packet
Parse-Head-X [+V -N T -Arg]

Pattern of new rule is:
   Cyclic node: Nil
   Active node: X-max [+V -N T -Arg]
   Buffer:
    [ -N T -Arg][ kiss +V -N +arg][ +N -V ]
```

> Modal

Target Rules 5: Progressive

Marcus version of rule:

{RULE PROGRESSIVE IN BUILD-AUX

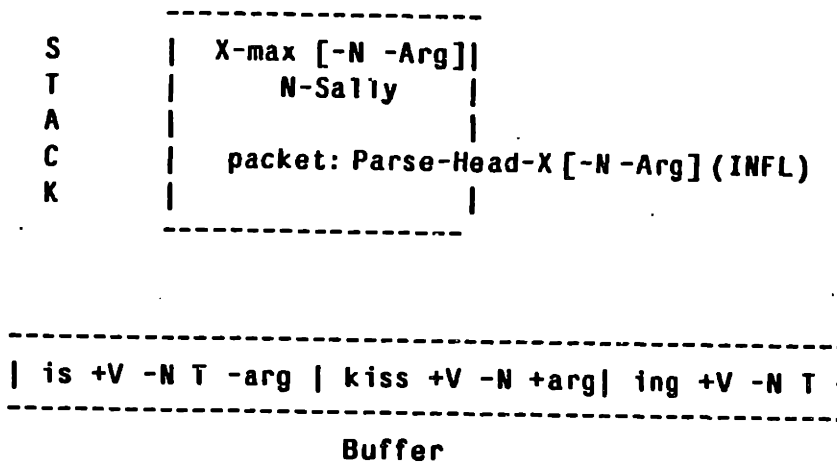
[=*be][=ing]-->
Attach 1st to c as progressive. Label c progressive.}

Target version of rule:

Cyclic node: Nil
Active node: INFL-max
Packet: Parse-Head-Infl
Buffer:
[be +V -N T -Arg][+V -N][ing +V -N T -Arg]

Example 1: Sally is kissing John.

Assume that *is* is known as [-N +V -Arg], from examples such as, *John is sick.* As usual, the Subject NP will be parsed, dropped into the buffer, and attached properly. The machine will be in the following state:



Note that the rule to attach *has* does not match in this case, because the third item currently in the buffer does not match the *have* rule, which calls for an *ed.* No known rules match, and the acquisition procedure is entered. The attempted *attach* works, since the unification check against the INFL node succeeds. The new rule is stored, and the current active node annotated. Observe that the pattern is the desired one.

Target Rule 6: *Do* support.

Marcus version of rule:

{RULE DO-SUPPORT IN BUILD-AUX

[=*do][=tensless]-->
Attach 1st to c as do.}

Target version of rule:

Cyclic node: Nil
Active node: INFL-max
Packet: Parse-Head-INFL
Buffer:
[do +V -N T -Arg][+V -N +Arg][+N -V]

Example 1: Sally did kiss John.

This example will clearly be analyzed just like the modal cases. In fact, since the left- and right-hand contexts are the same for *did*, *could*, and *will*, at least given these example, the acquisition procedure will place *did* in the same lexical equivalence class as *did* and *could* -- incorrectly. This flaw is revealed by examples such as, *Sally could be kissing John*, which has the following buffer pattern:

[could -N T -Arg][be +V -N -Arg][kiss +V -N +Arg]

Since *could* and *will* possess this insertion context as well as the one for *did*, whereas *did* never exhibits this insertion context, this example will split the class consisting of {did, could, will} into {did} and {could, will}.[70] This example provides a case where an over-generalized class is refined by later examples. One crucial point about this procedure is that the classes are formed in a data-driven fashion. If no examples are ever provided that distinguish *could* or *will* from *did*, then the over-generalized class is never refined. Therefore, if one wants to guarantee that the correct rule system will be inferred, one must also make the assumption that a "covering" sample of positive sentences is presented, namely, one in which all possible distinct transitions and states of the automaton representation of the rule system have been presented in the form of at least one example.[71]

There is one problem with this analysis, however. How is it known that *kiss* is in fact [+V -N]? Why couldn't it be (ambiguously) marked [+N -V] at the point in the parse where *could* is attached. It

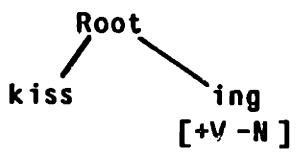
---

70. In practice then, lexical items are stored according to their insertion frames. A variant of this approach was in fact used by Marcus in the original Parsifal, where rules were hash-coded according to their first buffer element pattern.
71. See Chapter 3 for additional analysis of this point.

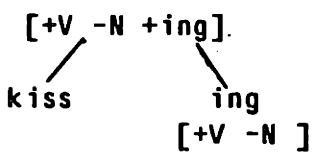
seems as if one must use the "ing" morphology to force "kiss" to be marked +V, rather than +N. Alternatively, one could make an additional buffer cell accessible to pattern matching, beyond the three now present.

This proposal amounts to saying that at the level of input analysis -- the level of phonological words, say -- *kissed* occupies one cell, while at the level at which attachment and movement operations occur, *kissed* can be broken down into two parts. There is some evidence that the morphological ending does not take up an extra cell, even though it is an abstract entity at the level of syntactic analysis. Consider Marcus' minimal pair, *Have the boys take the exam--Have the boys taken the exam.* Here, the crucial *ed* ending seems to be accessible to grammar rules dealing with *have*, even though the *ed* ending is beyond the three cell radius when *kissed* is broken into two parts. We conclude that a phonological word *kissed* only occupies one buffer cell, even though it can be dis-assembled via operations like *switch* or *attach.*

There is a slightly different way to view this proposal. Suppose that we adopt Lieber's idea that *kissed* is actually a miniature tree, consisting of stem+affix:



It is the root node of this tree that actually occupies a buffer position, even though this single nod is composed of two pieces. Note that this is exactly what happens when a buffer cell is filled by an entire constituent, such as an NP. Further, just as in the case of other maximal projections, we can assume that the root node of this constituent is annotated with features percolated from below (this is Lieber's proposal):



If it is the root node that fills a buffer cell, then this would mean that (i) *kiss* will be forced to be [+V -N], as desired; and (ii) the *ing* morphology will be "visible" to the rule that attaches *could.*

```
[could +V -N -Arg][be +V -N -Arg][+V -N  T +ing]
                                [kiss +V -N +arg][ ing +V -N  T -arg]
```

Importantly, the morphological ending will still be assumed accessible to grammar rule actions, such as *switch.*

This analysis allows the acquisition procedure to successfully distinguish *did* from *will.* As we will see immediately below, it is also necessary to successfully acquire a rule for passive *be.*

Target rule 7: Passive *be*

Marcus version of rule:

{RULE PASSIVE-AUX in BUILD-AUX
[=*be][=en]-->
Attach 1st to c as passive. Label 2nd passive.}

Target version of rule:

Cyclic node: Nil
Active node: INFL-max
Packet: Parse-Head-INFL
Buffer:
[was +V -N -arg][kiss +V -N +Arg +ed][*] (Note the collapse of kiss-ed into a single cell here.)

Example 1: Sally was kissed.

The analysis of this sentence proceeds as for the sentence, *Sally has kissed John.* There is one difference: the current sentence has no surface NP object following a Verb that takes an argument. No known rule matches the buffer pattern at the time *was* is to be attached; the *modal* rule calls for a Verb without an *ed* or *ing* ending (in the third buffer cell, or, in the collapsed cell system, in the second buffer cell). A new *attach* rule is built. Note that the current active node (INFL-max) is annotated with the name of this rule just executed; this annotation will prove to be important for the proper execution of a passive rule later on. The machine state stored with this action is:

```
Cyclic node: Nil
Active node: INFL-max
Packet:   Parse-Head-INFL
Buffer:
 [ was +V -N -arg][ kiss +V -N +Arg +ed][ # ]
```

Observe that this pattern is different from that currently known for *have* in *Sally has kissed John*, assuming now that *kissed* can be held in a single buffer cell:

```
Buffer for has attachment:
   [ has +V -N -arg][ kiss +V -N +Arg +ed][ +N -V]

Buffer for was attachment:
   [ was +V -N -arg][ kiss +V -N +Arg +ed][ # ]
```

Therefore, these two rules are <u>not</u> merged, as is appropriate in this case.

We have now acquired all the rules in Marcus' rule system for AUX, save for the Start-AUX rule, the AUX-attach rule, and the AUX-complete rule. But the start-aux and and compete-aux rules are handled by the automatic node creation and completion protocols. The third rule, the attachment of the AUX, is unecessary, since the maximal projection of INFL is actually the S node. If the S is an embedded clause, then there will be a later attachment rule for such clauses that will handle it. If the S is a root clause, nothing need be done.[72]

Before moving on to consider some finer points of the INFL system, we should point out why the acquisition procedure works correctly in these cases.

First, the use of the +Arg feature distinguishes verbs that subcategorize for NPs from those that do not. This prevents overgeneralization in certain crucial cases. For instance, the pattern for attachment of *could* in *Sally could kiss John* is different from that in *Sally could go*. As a result, the third buffer cell position is *not* generalized (and so reduced -- incorrectly -- to a "wild card" match, via the intersection of the features for # and the features for *John*). This is the right result.

Second, note that the system of rules so acquired rules out, e.g., the ungrammatical *Sally had kiss John*, because the pattern for *had* demands a *en* marker that is not present. There is however an apparent

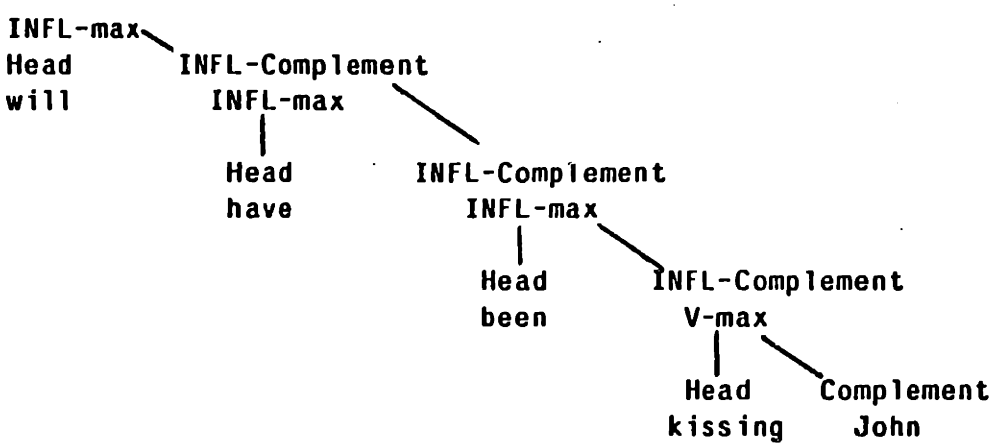
---

72. It is easy to see that the *to*-Infinitive rule will also be acquired, though this construction occurs only in embedded sentences. Presumably, *to* is known originally as an [-N] item. It is therefore compatible with the features of INFL, and is so attached.

problem with the ungrammatical example, *Sally will kissed John*. The problem is that rules trigger when features in the buffer meet all the conditions that the rule pattern demands. But the rule pattern for *will* says nothing at all about whether a *ed* pattern is required or not. Therefore, it seems to blithely handle *Sally will kissed John*, just as if it were *Sally will kiss John*. What has gone wrong?

While the buffer pattern meets the buffer rule trigger for the attachment of *will*, there is a difference when *kissed* is attached. For the active node (INFL-max) will be annotated at the time *will* is attached. This INFL node will be the current cyclic node at the time that *kissed* is attached as the Head of the VP, and part of the Complement of INFL. But the attachment rule for *kissed* does not have a left-hand context with this pattern, since the grammar rules to handle this item have appeared in sentences such as, *Sally has kissed John*, *Sally will have kissed John*, *Sally should have kissed John*, and the like. In the first sentence, the INFL-max node is annotated with the name of the rule to attach *have*; in the second case, the INFL-max (S) node is indeed annotated with the *will* attachment rule, but -- as we shall describe in detail in the analysis of sentences with multiple auxiliaries below -- that annotation is not part of the pattern for the attachment of *kissed*. Rather, it is the nearest INFL-max projection, based on *have*, that is part of the grammar rule pattern for *kissed* attachment. Thus, in either case, the known rules to attach *kissed* in *Sally will kissed John* fail. This is exactly the desired result.

As mentioned, the case of multiple auxiliary elements is an interesting one. It turns out that the system that is automatically induced by the acquisition procedure is like the "multiple verb" analysis of the Auxiliary, originally advanced by Ross [1967]; it even more closely resembles the analysis advanced by Baker [1981]. A sequence of auxiliaries is projected, naturally enough, as a series of INFL-max's, with the Complement of each INFL phrase being the next INFL phrase in line; the last one is the Complement position filled by the VP:



(See Chapter 3 for additional discussion.)

How are structures like this formed? Suppose an example like *Sally will have been kissing John* is encountered. The projection of the first INFL-max node and entry into its Complement portion are as before. Next, however, there is something new: *have* is projected as an INFL-max of its own. (The former active node, the INFL-max associated with *will*, is pushed down one in the active node stack.) As usual, *have* is attached as the Head of the new INFL projection, and the Complement portion is entered. History repeats itself; *been* triggers the creation of a new INFL-max, *been* is attached as its Head, and a Complement schema is entered. This time, it is the V *kissing* that triggers the formation of a maximal projection. *Kissing* is attached as its Head, and *John* as its Complement. The V-max is completed and dropped into the buffer, initiating a chain of phrase completions. The two INFL-max phrases are completed, and finally, the whole structure is attached to the main INFL-max. (Note that the Complement to the first INFL-max will be labelled with the feature +V -- percolated from the V-max below -- and so meet the NP-INFL-VP constraint.)

Because the features of the active node are annotated with the names of the rules that execute, local tree structure is encoded so that the ordering of the INFL system is recorded. For example, the order *have been kissing* is grammatical, *been have kissed* is not. What are the grammar rules built to handle a sentence such as, *Sally has been kissing John*? *Have* and *be* form their own INFL-max projections, with *kissing* a V-max attached as the Complement of *be*. Therefore, the grammar rule to attach *been* as the Head of a second INFL-max looks like this:

```
Cyclic node: INFL-max Rule Subject_NP_attach Rule Have_attach
Active node: INFL-max

Buffer:   [be +V -N +en][kiss +V -N +ing][John +N -V]
```

But then, the pattern of *Sally been has kissing John* does not allow *been* to be attached, since the left-hand context of this grammar rule will not match the left-hand context of the non-grammatical sentence at the relevant point.

Other cases of multiple auxiliaries are encoded in the same fashion; the details will notd be covered here. Again, for a formal analysis, see Chapter 3

This completes our discussion of the acquisition of the INFL system. Further examples of the acquisition of base rules, such as those required for Noun Phrases, Prepositional Phrases, the subcategorization frames for different verb complements, and embedded sentences, are outlined in the Appendix to this chapter. We will now turn to an analysis of the acquisition of non-base rules, including subject-auxiliary inversion, passive constructions, and simple *wh* movements.

2.6 Non-base Rules

2.6.1 Subject-auxiliary verb inversion

The target rule for Subject-Aux inversion is:

Packet: Parse-Specifier-INFL-max
Pattern:
Cyclic node: Nil
Active node: INFL-max (S)
Buffer: [auxverb][NP][*]
Action: switch

The effect of this rule is to switch the auxiliary verb into its normal declarative position. It is aimed at handling examples such as, Was a meeting scheduled? Will a meeting be scheduled? Did John schedule a meeting?

Note that inversion is not ordinarily permitted in non-root environments; hence the Nil value for the cyclic node position in the rule trigger. This is roughly correct, since inversion in embedded environments is not in general allowed:[73]

*I wonder was a meeting scheduled.

The acquisition of this rule was presented in general outline in Chapter 1. We can now understand the process in more detail. Suppose the following example is provided to the acquisition procedure after the INFL rule system has been acquired: *Will Sally kiss John?*.

Will is marked [-N T -Arg] and triggers the creation of an INFL-max, which becomes the current active node; the NP-INFL-VP schema is entered, with the packet set to Parse-Specifier-INFL-max, expecting an NP.[74] What next? The known grammar rules in this packet all attach a Subject NP to INFL-max; they do not match the current buffer pattern:

---

73. Baker [1979] observes some exceptions to this general rule, where inversion does not take place in a yes-no question.

How come he's still here?

These examples can probably be accounted for if one assumes that *how come*= an S something like, "why is it that". Then "he is still here" iis in an embedded environment, and inversion is prohibited, as desired.

74. Note that this assumes that *will* is not a Verb.

```
-----------------------------------------------------
|Will -N T -Arg|Sally +N -V|kiss +V -N +Arg (or -V +N -Arg)|
-----------------------------------------------------
```

(Note that the category identity of *kiss* is not yet determined as either N or V.)

The acquisition phase is entered. *Attach* is attempted, and fails the unification check, since the procedure's schema pointer is currently set to the "NP" portion of NP-INFL-VP. *Switch* is attempted next. Recall that in order for *switch* to work, it must exchange sisters dominated by the same node. In the case at hand, the second item in the buffer must be at least reduced to some phrase beyond a simple lexical token. Therefore, the acquisition procedure must first attempt to parse a complete constituent, starting in the second buffer position.[75] The easiest way to think of an attention shift is as the recursive invocation of the parser, but now with the left-hand edge of the buffer assumed to start at the second buffer cell position; the first item becomes temporarily "invisible." The parse of the second constituent is simple in this case; all the rules to analyze NP's are assumed to be known, so *Sally* is parsed as a complete NP as usual, and returned to the second cell in the input buffer:

```
-----------------------------------------------------------
|  Will -N T -Arg|  N-max  | kiss +V -N +Arg (or -V +N -Arg) |
-----------------------------------------------------------
```

Now the switch is performed. (Note that [+V -N -Arg]--[N-max] is *not* a known operator-operand structure, so this exchange can be made.) The buffer state after the *switch* is:

```
-----------------------------------------------------------
|  N-max  |  Will -N  T -Arg| kiss +V -N +Arg (or -V +N -Arg) |
-----------------------------------------------------------
```

Now the system checks whether a known rule will execute. The standard Subject_NP_attach rule in the packet Parse-Specifier-INFL-max matches; so the *switch* has succeeded. The trigger pattern for the new rule is the buffer pattern before the *switch* of course. The new rule is saved, and the current active node INFL-max, is annotated:

---

75. This is what Marcus called an "Attention shift." Formally, this method was suggested by Knuth [1965].

```
Cyclic node: Nil
Active node: INFL-max
Packet:   Parse-Specifier-INFL
Buffer:   [ Will -N T -Arg][ N-max][ kiss +V -N +Arg (-V +N)]
```

This annotation is required to distinguish between the active and inverted forms, a distinction useful for later semantic interpretation.

Note that this rule will *not* trigger in non-root environments; There the cyclic node will be non-Nil. Again we see that the procedure does not over-generalize, guessing too large a language.

So far, the inversion rule demands that *will* must be present as the first item in the buffer before a *switch* can take place. Further examples will generalize this rule: *Could Sally kiss John* sparks a grammar rule with the following pattern:

```
Cyclic node: Nil
Active node: INFL-max
Packet:   Parse-Specifier-INFL
Buffer:   [ Could -N T -Arg][ N-max][ kiss +V -N +arg (-V +N)]

Action: switch
```

The generalization protocol for non-*attach* rules applies. The left-hand and right-hand context of this rule is exactly the same as that for the previous *switch*, save for the first buffer cell items; so the features of the two rules are intersected to a get a new, generalized rule.

```
Cyclic node: Nil
Active node: INFL-max
Packet:   Parse-Specifier-INFL
Buffer:   [ T -N -Arg][ N-max][ kiss +V -N +arg (-V +N)]

Action: switch
```

Note that this process is unlike that used for *attach*, in that new lexical classes are not directly formed or split. In fact, the new generalized rule will also apply when a lexical item not in the same class as *could* or *will* is in the first buffer position, namely, *did*. This reflects the structural properties of Aux-inversion: the first auxiliary verb appears before the Subject NP, no matter what it is.

What though, about a sentence such as, *Has John kissed Sally*? *Have* can be either -Arg or +Arg -- and in fact, ought to be marked +Arg in this case, since the NP *John* appears after it is surface structure. There are two solutions to this difficulty. One -- not surprisingly -- is to appeal to the supplied predication representation. It will say that *John* is the Logical Subject of the sentence, that the predicate is *kiss*, and that the Argument of *kiss* is *Sally*. Hence, *John* is not the argument of *has*. The alternative for *has* -- its other categorization -- is as -Arg. This choice succeeds. Second, one might adopt a completely different analysis of Auxiliary inversion. Instead of saying that the leading Auxiliary verb must be switched into the proper position, one could just leave it in place as an operator whose complement is the entire remainder of the sentences -- an S:

```
Did -- [Sally kiss John]
Opr -- Operand

Has -- [Sally kissed John]
```

This view has some appeal, particularly when one extends the S rule to cases of embedded sentences, where *that* and *for* act like operators governing an entire S. (See immediately below.) This approach has not been implemented, but it is easy to see how it would work. The leading INFL element would be projected as INFL, as before. Then, instead of entering the Specifier portion of a special NP-INFL-VP schema, we would attach the INFL as the Head of the maximal projection just created. The remaining elements in the buffer would then be parsed as an S (with appropriate modifications to deal with the lack of tense in some cases).[76]

Negative adverbial sentences, *Never have I seen such a mess!* are discussed later in section 2.6, after complementizers have been introduced.

2.6.2 Passive Constructions

The target rule for the basic passive construction is:

Pattern:
Cyclic node: INFL-max Rule Attach_be
Active node: X-max [+V -N +arg]
Packet: Parse-Complement X [+V -N +arg]

---

76. The ambiguity in the categorization of *have* could result in the often-noted Have main verb-Subject NP inversion found in some dialects.

Buffer: [*][*][*]

Action: drop trace

This grammar rule is sufficient to handle cases such as, *Sally was kissed, Sally was kissed by John.* The effect of the passive rule is to drop a "dummy" NP node into the first cell of the buffer. Later rules will then attach this NP to its position in the Verb Phrase complement -- for example, as an Object Noun Phrase. Thus the purpose of this rule is to construct a parse that analyzes a passive sentence as a case of a trace-marked Noun Phrase, with the trace bound to the now displaced NP. For example, in the sentence,

Sally was kissed.

the Noun Phrase *Sally* is really the Object argument of the verb *kissed* for the purposes of interpretation -- just as it is in the straightforward declarative version of the same sentence, *John kissed Sally.* The trace serves to express this relationship, indicating where the NP's "true position" in argument structure:

[Sally]$_i$ was kissed [*trace*]$_i$.

For this rule to trigger correctly, it is crucial that the cyclic node be marked with the feature "+ed" and the active V-max node be marked "+be", corresponding to the pattern for English passives,[77]

NP be V +ed

The assembly of rules to deal with the passive construction illustrates some interesting points about the acquisition procedure itself. As we will show, because the system is designed to be conservative and incremental in its rule formation, building new rules only when the inference is clear-cut. As it turns

---

77. Note that this pattern will also trigger on Np-be-Adjective +ed constructions, such as, *The boy was unlearned*, unless the restriction +Arg is included in the rule pattern. But this is precisely what is known, since *unlearned* never appears with a surface direct object: **John unlearned the boy*. Thus, the subcategorization frame for *unlearn* will not include a direct Object, and the drop trace rule will never be built to handle such cases. Rather, *unlearned* will be attached as an Adjective, just as in the comparable case, *John was sick*. As a result, the drop trace rule will never be (over)generalized, and always retain its initial "+Arg" marker in its triggering pattern. Alternatively, this distinction could be encoded by means of distinct packet names: a verb that demands a direct object could activate the packet Parse-Object, but a predicate adjective that cannot take a direct object could activate a different packet. The passive rule is cached in the packet Parse-Object, and so would never be triggered by the predicate adjective construction.

There are other auxiliary verbs in English that allow passive (Baker [1979]), for example, *got*:

John got arrested.

The acquisition procedure also acquires this alternative construction.

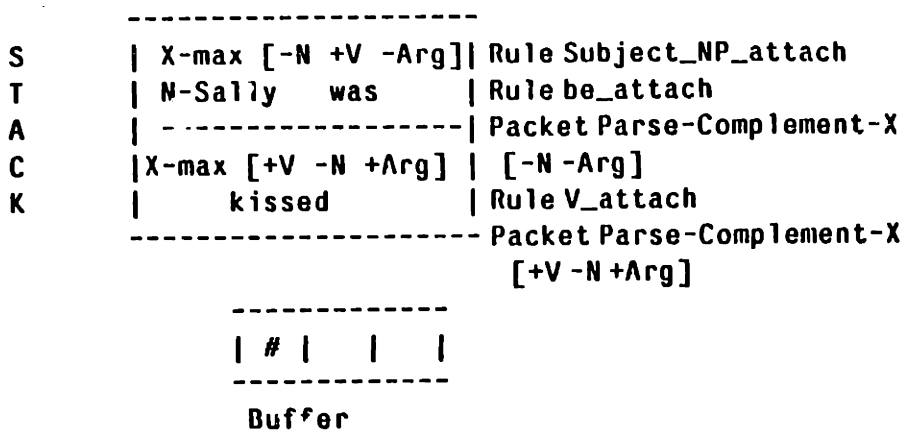
out, this has the effect of making the acquisition of so-called "truncated" passives, passive constructions where there is no *by* phrase, as in, *Sally was kissed*, easier than the acquisition of full passives -- where there is a *by* phrase, as in *Sally was kissed by John*. The basic reason for this is two-fold: (i) the absence of an Object NP is plainer in the truncated passive case, so that the linking of the NP in Subject position to the argument of the V predicate is clearer; (ii) analyzing a full passive requires an ability to handle *by* Prepositional Phrases, with all that entails. For both of these reasons, full passives are "more complicated" than truncated passives. Not unexpectedly then, the acquisition of full passives occurs later than for truncated passives, both in the acquisition procedure and in children. Simple declaratives are understood and produced first.[78]

Let us turn directly to an example. Given that the rules to handle simple declaratives are known, suppose that a truncated passive is presented to the acquisition procedure: *Sally was kissed.*

The analysis of the first three tokens proceeds normally. The NP *Sally* is attached as the Specifier of INFL-max (S); *was* is attached as the Head of INFL, annotating INFL-max in the process. A V-max phrase is formed, and *kissed* attached as its Head. Note that *kiss* is forced to the features pattern [+V -N +Arg], since the alternative, *kiss* as [+N -V], is not compatible with the NP-INFL-VP schema. Since the Head has been attached, the Complement packet of the X-max [+V -N +Arg] (VP) currently active is entered. The feature "+Arg" -- the proxy for the subcategorization frame for *kiss* -- indicates that an NP must follow. (Note that this fact crucially makes use of the Projection Principle.) No known rules match, because in fact the only rule known in the Parse-Complement-X [+V -N +Arg] is the rule to attach an Object NP to the Verb Complement. That is, the subcategorization frame for *kiss* has not been satisfied. This blocks completion of the phrase; hence, the V-max phrase is cannot be dropped into the buffer. The acquisition procedure is invoked. At this point the machine state is as follows:

---

78. See Limber [1973]; Maratsos [1978]. For the latter source and an analysis of the acquisition of passive within the Government-Binding theory, I have relied on a paper by A. Weinberg [1982].



Attach is attempted, but fails, since the end-of-sentence marker # can never be attached to any node. Note how clear-cut this case is. Likewise, *switch* has nothing to *switch*, and so cannot apply. *Drop trace* is next. An empty NP element is inserted into the first position in the buffer. Now, can any known grammar rule execute? Yes: the grammar rule for attaching NP Objects looks like this:

```
Cyclic node: X-max [+N +V -Arg] Rule NP_attach
                Packet Parse-Complement-X

Active node: X-max [+V -N +Arg] Rule V_attach
                Packet Parse-Complement-X

Buffer:   [ NP ][ # ]
```

Crucially, this rule *does not say that the be_Attach rule must be present*; therefore, the predicate demanded by this rule pattern triggers against the current machine state. (Recall that rule patterns are predicates that must be true of the current parser state before the associated rule action can execute.)

Therefore, the *drop trace* rule has succeeded. The new rule pattern is saved. Observe that now the rule pattern for passive will include the triggering predicate "be_attach" annotation on the X-max node above the current active node, since that feature was present at the time the new rule was built. The VP node is annotated with the name of the new passive rule, again to provide a basis for potential semantic differences, if necessary. The remainder of the parse will proceed normally.

Nothing has been said about how proper thematic assignment is made in this case. *Sally* is actually the argument to the predicate *kissed*, just as *John* is the argument to *sick* in, *John is sick*. How can this be determined? Suppose we assume that every dummy NP (NP trace) must be bound to some antecedent NP. Further assume that this antecedent must be accessible to the dummy NP. Observe first that

because of the structure of the parser at most one cyclic node can be interposed between the NP trace and its antecedent; more distant nodes are simply not visible. Thus the parser directly simulates the effect of the Subjacency constraint on movement.[79]

In the case at hand, the only possible NP antecedent is *Sally*. If the NP trace is bound to this NP, then *Sally* will become the argument to the predicate *kissed*, assuming that the rules that interpret arguments to predicates are held constant.[80] Note that if predicate-argument information is assumed given, then this hypothesis can be confirmed. What of the rule (tacitly assumed so far) that interprets the Logical Subject as the Agent of the sentence? We now have a situation where *Sally* is linked to the argument position of the predicate *kissed*, and that move is confirmed by (assumed given) independent evidence. But the assignment of an Agent thematic role to *Sally* is no longer on safe ground, assuming that it is somehow known that *Sally* is not the Agent of the situation. Thus we have the following facts: *Sally* is linked and interpreted correctly as the argument to *kissed*; the NP *Sally* is not the Agent of the sentence, even though it appears in the proper position to be interpreted as such. This is a case where the system of rules that assign thematic interpretations should be changed. The acquisition of this knowledge has not been the subject of this research, but something can be said here. Just as the failure of grammar rules stimulate the formation of new rules in the operation of the parser, it seems plausible to say that the failure of rules of thematic assignment could be used to prompt the formation of new thematic mappings, or even new thematic categories. This is the case here. *Sally* appears in Logical Subject position, yet is the argument to *kissed*, and is not an Agent. Suppose we say that when an NP is in a structural Subject position, but is an argument to the main predicate of the sentence, then it is the theme of the sentence. Thus *John* is the theme of *John is sick*. Interestingly enough then, in this case one would derive the "concept" of a new thematic role from the structural relations in the passive sentence, plus existing knowledge about very simplethematic roles -- in contrast to the approach that would derive a notion of passive constructions from thematic relations alone. Though this is just the sketch of a proposal for the acquisition of thematic role mappings, it appears plausible, at least in this case, and is worth further study.[81]

What about full passive sentences? They require a prior ability to analyze Prepositional Phrases. At a minimum this means the ability to recognize *by* as a Preposition assigning some kind of thematic role

---

79. This is a different approach than that taken in Marcus [1980]. Marcus [1980] attempted to derive Subjacency from other constraints of the parser. He pointed out, however, that one could directly encode Subjacency in the way just described.

80. In this sense, the Projection Principle ensures semantic transparency: a slighly abstract syntactic form preserves semantic interpretation. One might, in contrast, take *syntactic* transparency as one's goal, and give up the criterion of semantic transparency. For example, instead of arguing that the surface form, *I want to leave* is not what it appears to be, and actually has canonical NP-INFL-VP structure, something like [I want [I to leave]], one might insist that the surface syntactic form is exactly what it appears to be, namely, [I want [VP to leave]], and complicate the system of semantic interpretation. This seems to be the basic thrust of Lexical-Functional Grammar [Bresnan, 1982].

81. Moreover, there is evidence children acquire so-called non-reversible passives -- sentences such as, *The plant was kissed* -- even earlier. (See Weinberg [1982] for additional discussion.)

to its Object NP. It is not surprising then to discover that children often mis-analyze such sentences, particularly in reversible passive sentences where the Subject or Object NP can be legtimately exchanged. (*John was kissed by Sally*)

The acquisition procedure mimicks this developmental staging effect. For For suppose that grammar rules have not been acquired to handle Prepositional Phrases in Verb Complements. Then, even after an NP trace is inserted in such sentences as, *Sally was kissed by John*, no known rules will execute. The system must wait until it has acquired the ability to parse PP's before going on to handle full passives. Therefore, there will be a tendency for truncated passives to be analyzable before full passives.[82]

In any case, suppose that PP's can be successfully analyzed and that the system receives the example, *John was kissed by Sally*. At the point where the NP trace should be dropped, the buffer looks like this:

```
-----------------------------------
| by -N -V +Arg | John +N -V |  #  |
-----------------------------------
```

No rules execute. *Attach* fails because *by* fails the unification test for an NP Complement to *kiss*. *Switch* fails because an exchange would destroy Operator-Operand order; here is another place where knowledge of PP's must be used. *Drop trace* works, if we assume that sentences with ordinary NP Complements followed by PP Complements have been seen before, e.g., *John kissed Sally with passion*, *John kissed Sally behind the school* results in the proper grammar rule pattern to handle the case above. Given all of these prerequisites, then the *drop trace* works, and is saved as a new rule. Since it is the same action and left-hand context the truncated *drop trace* rule, by the protocol for mergin non-*attach* actions, we intersect the features of the items of the two buffer patterns, obtaining:

```
Cyclic node: INFL-max  Rule be_attach
Active node: V-max   Rule kiss_attach (subcateg. frame [___NP])
Packet:   Parse-Complement-X [+V -N +Arg]
Buffer:   [ * ][ * ][ * ]
```

This is exactly the target result.

---

82. Note that it does not necessarily follow that truncated passives will become analyzable inevitably before full passives, just that it is more likely that this will happen. If a sequence of PP Complement examples is given, then the system could acquire these rules before seeing any passives at all.

One final remark. An important question to ask is why, having dropped an NP trace, the system does not immediately fire a passive rule once again. For the input buffer now matches once again the pattern of the passive rule just acquired. The annotation of the active node does not help here, so the passive rule will just execute again. If left unchecked, this possibility would lead the machine into an endless loop. Recall however that there is a default priority ordering of grammar rules: if two rules in the same packet match the features of the current machine state, then the most specific rule gets priority. In the case at hand, there is another rule that can execute, namely, the rule that attaches NP objects to the V-max. But this rule demands a more specific pattern match of the input buffer:

[NP][*][*]

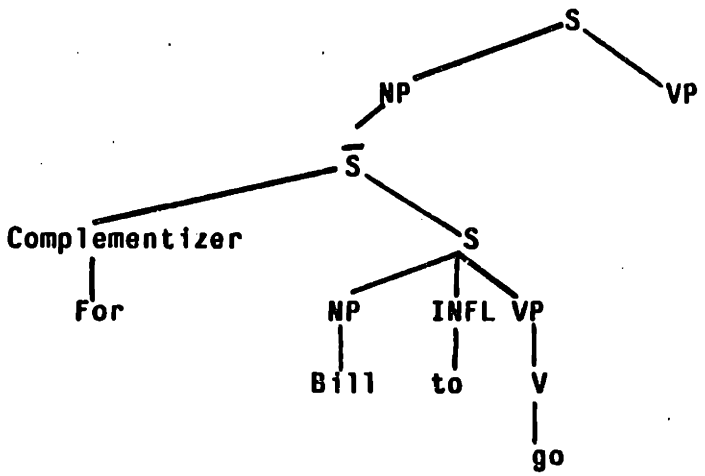
Thus, the Object-attach rule takes priority over the passive rule, attaching the NP trace to the V-max. Now the buffer has the end marker # in its first cell, and the V-max has been annotated with the NP attachment rule. Now the subcategorization frame for *kiss* is satisfied, and the completed V-max can be dropped into the buffer, deactivating any rules associated with the V-max-Complement packet. The remainder of the parse proceeds as in a simple declarative sentence.

2.6.3 Verb Complements and Embedded Sentences

So far, we have not had to expand the X system to include embedded sentences, either as Subjects, Objects, or relative clauses:

For Bill to go would be foolish.
I promised *John to go.*
The man *that left early* is sick.

The usual analysis of the first two kinds of sentences extends S to a higher level bar category, $\bar{S}$, adding a "Complementizer" in the form of *for* or *that* (see Bresnan [1972]):



Note that the structure of S̄ follows the operator-operand (Head-Complement) order fixed by other examples in English, if we consider *for*, or *that* to be operators. This is a natural assumption. *For* assigns Case to its NP argument in Prepositional Phrases; *that* is a deictic operator. Moreover, this choice makes the Complementizer the Head of S̄.[83] One advantage to this is that it makes *for* the governor of *John* in sentences such as, *For John to leave would be terrible.* Then all cases of government fall under a canonical operator-operand pattern, the core notion of government: Preposition--NP; Verb--NP or S̄; and Complementizer--S.

Additionally, some theories hold *Wh* elements also fill the Complementizer position, intuitively, acting like operators:

Why--did Sally kiss John?
Opr--Operand

How--will you get home?
Opr--Operand

Who--will Sally kiss?
Opr--Operand

How is this extension of the X̄ system acquired? The *order* is fixed by other examples, as mentioned. An example will show how the remainder of the S̄ system can be fixed.

Consider an example such as, *Sally got a candy for Bill to take.* Assume that *got* has already been

83. S̄, recall, is the maximal projection of INFL.

successfully analyzed in other constructions, so that it is known that it takes an argument (subcategorizes for an NP or an $\overline{S}$). Assume further that *for* has been analyzed in PP constructions, so that it is known as an Operator in Operator-Operand configurations.

The parse proceeds smoothly until the point where the packet associated with the analysis of the Complement of the Object NP is activated. At this point, the parse looks like this:

```
          ---------------------
S         | X-max [-N +V -Arg]| Rule NP_attach
T         | N-Sally           | Packet Parse-Complement-X
A         |-------------------|  [-N -Arg]
C         |X-max [+V -N +Arg] | Rule V_attach
K         |     got           | Packet Parse-Complement-X
          ---------------------  [+V -N +Arg]
          | X-max [+N -V -Arg]| Rule A_attach Rule N_attach
          |  a    candy       | Packet Parse-Complement-X
          ---------------------  [+N -V -Arg]

              -------------------
              | for | Bill | to |
              -------------------
                    Buffer
```

Next, *for* is projected as an X-max, since it is known as an operator (a Head item). Since *for* is the Head of this phrase, it is attached as such; the Specifier packet is skipped. The Complement portion of the analysis of the *for*-phrase is entered. (Recall that Specifier-Head-Complement order was assumed to be established.) The +N lexical item prompts a parse of an NP via an "attention shift," again as usual, returning an NP to the first cell in the buffer; we ignore the details of this analysis here.

If the NP *Bill* were simply the Complement of a PP phrase *for Bill*, then at this point the grammar rule to handle this construction would fire. But this rule triggers on buffer patterns of the form,[84]

```
-------------     ----------------------
| NP | # |  | or | NP | Prep | +N -V |
-------------     ----------------------
```

84. The patterns below are kept separate because the corresponding action is *attach* and the k-tails of the buffer patterns are distinct.

(E.g., I went with Bill to school.)

But the current buffer looks like:

```
--------------------------------
| NP | to -N  | take +V -N +arg|
--------------------------------
```

Therefore, existing grammar rules do not trigger. However, the S-creation rule *does* trigger: the buffer is in the form NP--+INFL (OPR)--+V. An X-max of this type is created and placed on the active node stack. The NP is attached as the Specifier of the phrase. What about *to*? It is known to be -N, an Operator, and, in this case, does not have an NP in surface structure immediately to its right. Therefore, it meets all the conditions to be attached as the Head of the S (its features are compatible with INFL-max).[85] Following the usual invocation of the acquisition procedure, a new attachment grammar rule is created to carry out this operation.

With Specifier and Head attached, analysis of the complement of the X-max (the S) is now attempted. The lexical item *take* prompts the creation of yet another X-max, the VP. *Take* will be attached as the Head of this X-max, via already known rules. This leaves the buffer with # in the first cell. However, no known rules execute; the grammar rules to handle # expect a left-context consisting of either an annotation on the VP node of -Arg and no annotation via the Rule Object_NP_attach (i.e., an intransitive verb), or else an annotation of +Arg and Rule Object_NP_attach (a simple transitive verb, with subcategorization frame satisfied). Neither of these conditions are met by the current left-context, which has the annotation +Arg on *take*, but no annotation via Rule NP_attach, indicating that the NP Object has not yet been attached to the parse tree. Therefore, the acquisition procedure is entered. *Attach* and *switch* clearly fail. *Drop trace*, however, succeeds, just as in the passive case. Omitting details of how trace binding works in this case, the NP is attached, and the current active X-max, the S, is now complete. Like any completed phrase, it is dropped into the input buffer, uncovering the *for* X-max as the current active node.

We are now back to the point where the Complement portion of the *for* phrase is under analysis. No known grammar rules trigger in this situation. The acquisition phase is entered. *Attach* is considered first. The X-max is an S; hence is assumed known as a +Arg item. *For* is looking for a +Arg Complement; hence the attachment passes muster, and a new grammar rule is built. Note that this

---

85. Note that *to* does *not* have the feature +Arg in this case, as it would in the construction, *to Bill*. It is also non-predicative, just as the modals *should* and *might* are.

phrase is embedded within an NP:

```
Pattern:
 Cyclic node: X-max [+N -V]
 Active node: X-max [-N -V +arg]
 Packet:   Parse-Complement-X [-N -V +arg]
 Buffer:   [ S ][ # ]

Action: attach
```

With the *for*-phrase built, it too is dropped into the buffer, and attached as the Complement of the NP by yet another new grammar rule:

```
Pattern:
 Cyclic node: X-max [+V -N -arg] (INFL-max)
 Active node: X-max [+N -V -arg]
 Packet:   Parse-Complement-X [+N -V -arg]
 Buffer:   [ X-max -N -V +arg ][ # ]

Action: attach
```

Finally, the entire NP is now complete; it is dropped and attached as the complement of the VP by a known rule.

In all, three new rules are acquired: (1) The expansion of S as Operator-S; (2) the attachment of *to* as INFL; (3) the attachment of S as the complement of an NP. In addition, in this case a rule to handle a relative clause gap is acquired.

What evidence is there for this analysis? There are some suggestive indications from child language production that the Operator status of *for* as a Preposition triggers its use as a Complementizer. Roeper [1980] has collected examples of sentences produced by a child 2.5 - 2.10 years old that show that *for* is indeed used in for-NP constructions quite early:

> This one is bigger for you.
> There's one for Mom for to brush he's too.

Evidently, children of this age recognize *for* in the input stream of sentences and can use it productively. It is not surprising, then, that for-S constructions are also found:

This is for I can put some light on.
I want for you hold it.
It's too big for you eat.
It's for fix things.

Note that the *to*-inflection is often dropped, as is often the case with inflectional elements generally at this age. More interestingly, as Roeper observes, these early uses of for-S constructions obey a clear semantics that is parasitic on that used in *for* Prepositional forms. Namely, the use of *for* as a purposive, as in, *I want this for Mom*, portends its use as a purposive in S's, e.g., *It's for fix things*. This is then another example of how syntactic/semantic conflation drives acquisition. In both cases, the argument status of the trailing phrase is somehow apparent, and a thematic role can be assigned. But because the +Arg status of the S is clear, it can serve as a Complement. Once this initial example is used to fix the syntactic extension, then the requirement that the for-S phrase be purposive can be relaxed.

Roeper's sample also includes many examples of *for-to* constructions:

Let's bring a bench for to jump in.
The milk is for to drink.
Toys are for to play with.

Why do these sentences appear? If *for* is known to be *required* as an Operator, assigning some kind of purposive thematic role, then it must be present, at some level of representation. Since there is no known rule to block its appearance in forms like, *Toys are for to play with*, it will appear. Note that this surface form is not permitted in most English dialects, but that the underlying representation of *Toys are to play with* is the child's version. Thus the child's sentence is simply a "purer" representation of underlying thematic structure -- as expected. Moreover, sentences such as *The milk is for to drink* will be parsed perfectly well by a (hypothetical) self-analyzer that subjects sentences that are to be generated to error checks; a trace will simply be dropped to fill the role of the NP that is missing.

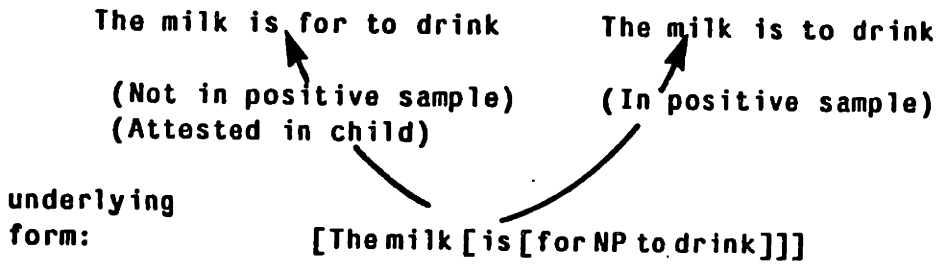
Finally, there is evidence that the full *for*-S construction is acquired by this age:

It's for Daddy to look at.
Som berries for the birds to eat.

How could the adult *for-to* filter be acquired? If the child is in a dialect community that uses *for-to*, then of course it will hear such constructions and be able to parse and generate them; all is well. But what of the case where the community does not use *for-to*? As we have seen, the child still *generates* such cases, early on. This would seem to be an explicit violation of the Subset Principle: the child has guessed a language that is "too large," and there is no positive evidence that it could receive that will

tell it that this is incorrect.

There is a way out of this dilemma, however. Suppose that we invoke a corollary of the Subset Principle, Wexler's Biuniqueness Principle (see Lasnik [1981]). The Biuniqueness Principle (or Uniqueness Principle, since we will need only one-half of Bi-uniqueness) is a constraint on the evaluation procedure for grammars such that the learner always maintains at most one surface form for every corresponding deep structure. Note that this is indeed a corollary of the Subset Principle. Its effect is to narrow the class of possible output surface forms. In the case at hand, it would permit the existence of forms such as, *The milk is for to drink* along with, *Grandma has a present for me to blow on*, as long as there were no other alternative surface forms corresponding to the deep structures for these sentences. However, as soon as a positive example surface string such as, *The milk is to drink* can be analyzed, the Bi-uniqueness Principle would block the generation of *for-to* from then on, because there would be an alternative surface form corresponding to this one underlying structure. The *for-to* surface form need not be used:



A model of generation of this kind, along with appropriate modifications to handle self-correction and the acquisition of filters as suggested by the Bi-uniqueness Principle, has not been considered in this thesis. An extension of this kind would be an interesting direction for future work.

Turning now to other S-complement structures, consider first *that* Complements, e.g., *Sally knows that Bill kissed Jane*. The parse of such sentences proceeds normally until we reach a point where the input buffer holds the following items (we assume that *Bill* is analyzed as an NP, as usual):

| that | Bill | kissed +V -N +Arg|

Now we have a choice. *That* can be projected as an X-max of some sort, or it could be the Specifier of an X-max (as in, *that boy*). Suppose the latter course is selected. There is however nothing to attach *that* to; there is no Head item to prompt the formation of an X-max of the right sort (assuming that

Bill is known as a complete NP).[86] Unification fails if this option is selected, so we take the first choice and project *that* as an X-max.[87]

Supposing that the first option is selected, then *that* is projected as an X-max phrase, with *that* attached as its Head; the Complement portion of the analysis of this phrase is then entered. As always, *Bill* will be parsed as a complete NP and returned to the input buffer. Then, as before, the NP--OPR--V configuration triggers the creation of an S. (In this case, S-creation is also prompted by the projection of the inflected verb and operator *kissed*.) This maximal projection becomes the new active node. The remainder of the parse proceeds just as in the parse of a root sentence, with one exception: wherever previous grammar rules specified a cyclic node of "nil", corresponding to a root S, new grammar rules must be acquired with a cyclic node pattern containing an X-max (an INFL-max, or S). So for example, the rules to attach an NP as the Specifier of an S, INFL as the Head of S, and V and NP to the VP, are all acquired with this modified left-context.[88]

Why is the form *Sally knows that Bill kissed Sally* permitted, but not the corresponding tenseless form, *Sally knows that Bill to kiss Sally*? One explanation for this difference is that the NP *Bill* must be "Case marked" in order to appear in a surface form, and that Case marking is performed by a Tense element or a Preposition. *That* is neither, and so the sentence that *Bill* is in must be tensed in order for it to pass this test. In contrast, since *for* assigns Case, *Bill* can appear in a tenseless clause after *for*, as in, *I want very much for Bill to win.* This property is *encoded* by the acquisition procedure -- the grammar rules that are built to handle embedded S's in *that* clauses have patterns that trigger on tensed V's, not tenseless V's. Therefore, tenseless V's in *that* clauses will not be directly parsable, the right result. Note that the parser left-context distinguishes *that* from *for* clauses, since the X-max corresponding to S will be annotated with the name of the rule that attaches either *that* or *for*.

More complicated complement structures can now be analyzed. Consider a verb that takes an NP and S complement, e.g., *persuade*, as in, *I persuaded Bill that John should kiss Sally*. Suppose that examples where *persuade* takes just an NP have already been encountered, e.g., *I persuaded Bill*. Then *persuade* will already be marked +Arg. What happens when the system attempts to parse *I persuaded Bill that he should kiss Sally*? As in other examples, the parse proceeds normally to the point where

---

86. Note that in an example such as, *That boy kissed Sally*, *boy* will trigger the formation of an X-max that can have a Specifier attached to it. In contrast, if this option is selected for *that Bill*, unification fails.

87. Alternatively, if it is assumed that a predicate-argument representation can be independently reconstructed, then the procedure would know that the argument to *know* is a full S rather than the NP *that Bill*: know (Bill kissed Jane). Then entry into the Complement of the V could be used to trigger creation of an S projection of some kind. The Complement could be set up to demand the features "+Propositional," or something like it, rather than "+NP."

88. Should grammar rules with nearly identical left-most contexts be merged? According to automata induction theory, non-identical left-contexts correspond to non-identical states. In general then, grammar rules with non-identical left-contexts should not be merged.

the Complement of the V is being analyzed. *Bill* is then parsed as a completed NP, perhaps drawing on extra-linguistic information. Now no known grammar rules execute; the NP_attach rule triggers on patterns like, NP-# or NP-Preposition-+N, not NP-that-+N. The acquisition procedure is entered, and an *attach* succeeds, since *persuade* demands a +Arg complement. At this point, the buffer contains the following items:

```
-----------------------------------
| that  | John +N+-V | should +H -N|
-----------------------------------
```

Now, *that* is known as both a possible Specifier of NP (a +N -V -Head item) and as Head of S, (-N -V +Head). As before, only the choice of *that* as a Head will work, since attachment of *that* as a Specifier will fail.[89] A new X-max is formed and becomes the current active node. Now analysis proceeds exactly as in the case of *know that -S. That* is attached as the Head of this phrase, and Complement analysis is initiated; the NP-V combination prompts the creation of an X-max of type S; the NP and VP are analyzed without incident.

One problem that remains with such examples concerns the representation of subcategorization information. *Persuade* takes an NP and a Propositional argument, and the system must be able to represent this fact so as to block, e.g., *John persuaded Bill to Sam.* So far, we have only allowed the representation of a single NP or S argument to be stored in the left-context of a parse, in the form of the "+Arg" feature. This feature is checked via unification when an element is attached to the maximal projection of which the Verb is a Head. For example, *persuade* is stored along with its associated local context predicate (its subcategorization frame) [NP, S]; *give* store the context [NP, *to* NP].[90] This context is used as a complicated "feature check" on the active node to which NP arguments are attached. Nodes violating the predicates are ruled ungrammatical. For example, suppose that *persuade* is projected as a V-max (a VP), and that two NP's are attached to it, as in, *I persuaded Bill to Sam.* Then the "features" of the VP now include [NP, *to* NP], whereas *persuade* demands the features [NP, S]. Thus the example would be thrown out -- even though it is perfectly well "parsable," in some sense. Note finally that the order NP-S for *persuade* is fixed by the learner via positive examples. The adjacency requirement for NP arguments is determined by cases such as, *I persuaded John, I kissed Sally*, and the like. This requirement is never weakened, at least in English. See Chapter 3 for additional discussion of this adjacency constraint in the context of the Subset Principle.

---

89. Again, if *that* is assumed to be a Specifier, then there will be nothing to attach it to.
90. For details on Dativization, see Chapter 3.

What of complement sentences with "missing" Subjects, e.g., *I want to go, I expected to go*? To see how these constructions are acquired, suppose that *want* has been analyzed as *want--X*, where X is an argument.[91]

Again, the parse proceeds without incident until the Complement portion of the V-max is entered. The buffer holds the lexical items *to, go* and #, respectively. Following the node creation protocol, we first project *to* as an X-max. Since *to* is followed by the +V item *go, to* has the features of a modal items in this context; it is an Operator (and is marked +Head); it is -Arg (not immediately followed by an NP argument); it is neither Substantive nor Predicative. To meet the subcategorization demand of *want*, this X-max can be either an NP or an S. Suppose we choose it to be an NP. Then *to* is the Head of this phrase, a contradiction since *to* is -N. Therefore, this choice fails unification. Suppose then that the X-max is identified as an S. Then the packet system for S is entered -- i.e., the constituent order NP--INFL--VP is expected.

Following the expected X order for S, we first activate rules to parse Subject NPs. However, none of these rules trigger. *Attach* fails because *to* is marked as the Head of the S, and is -N; *switch* fails because Operator-Operand order would be destroyed and also because *go* is marked -N. Finally, *drop trace* is attempted. It succeeds, since there is a rule in the Subject NP packet to handle *I want Bill to go*, by assumption. Ignoring for now the question of binding the trace -- trivial in this case since there is no possible antecedent besides the Subject NP in the higher clause -- the parse proceeds from this point just as if the sentence *I want Bill to go* were being analyzed.[92]

Why should children be able to produce *John wants to go* before they can produce *I want Bill to go*? The evidence is difficult to assess (as usual), but in part it seems as if the thematic structure of *I want to go* is relatively clear-cut, compared to forms with an explicit lexical NP, such as *I want Bill to go.* Note that in sentences with an explicit lexical NP that the possibility for mis-interpretation is present: *Bill* might be parsed as the Object NP, if, for some reason, the elements in the input buffer are neglected. Speculating, it could well be that attentional deficits limit the input buffer that children have, so that the third element, *go*, is not even seen. Since an Object NP is a legitimate alternative choice, confusion

---

91. It has been observed that sentences such as *I want to go* seem to be produced by children before corresponding sentences with full lexical NP's, e.g., *I want Bill to go.* This would seem to run counter to an assumption that *I want Bill to go* be analyzable before *I want to go* is. However, the situation is quite complex here, because of the uncertain status of complements in *want-X* constructions. It could be that *to go* is an X-max of uncertain categorial status, but that it has a kind of quasi-NP status -- it is *something* that is wanted, but not a physical object, rather, a state of affairs. Note that propositions have this quasi-NP status as well. If this is so, then *to go* would simply be analyzed as an X-max with an ill-defined categorial identity, but one at least fulfilling the subcategorization demand that *want* have some kind of Argument complement.

92. We also ignore for the moment the question of whether this null Subject is a trace or a pronominal-like element (PRO). Note that because lexical NP's are allowed in this position that the element must actually be PRO, not trace; the antecedent NP is also in a position with a real thematic role, as is never the case with antecedents of NP traces. (Compare: *I was kissed by Sally -- NP-trace.*

can arise. In contrast, a sentence without a lexical NP cannot be so mis-interpreted; the Complement *to go* cannot be an Object NP, since it is not an Object. Whether the Logical Subject of this phrase is supplied via the correct syntactic analysis or whether it is supplied via some sort of interpretive procedure -- also possible in this clear-cut case, since there are no alternatives -- the sentence with a missing Subject is actually "simpler" according to the criterion of syntactic-semantic correspondence. It is this transparency that could explain the earlier appearance of missing Subject Complements.

Once examples such as *I want to go* have been successfully parsed, then because of the way lexical equivalence classes have been established, other verb types will now be analyzable. The details of the process are interesting, since they show how verb classes may be split. Consider the example, *I tried to go*. So far, *want* and *try* have been placed in the same equivalence classes. Both take Argument Complements: *I want the candy/ I tried the candy*. Therefore, the analysis of *I tried to leave* will follow that of *I want to leave*: a dummy NP will be inserted as the Subject of the embedded clause following *try*.

This is, of course, not quite the right analysis -- or rather, we have not yet advanced any representational machinery to even state the right analysis. In the system as it stands all empty categories are alike; there is no difference between the empty category that serves as the Subject NP in *I want to win* and the empty category in *I tried to win*. There is some difference between *want* and *try* of course, since the former verb, but not the latter, allows a lexical NP in its embedded S: *I want Bill to win* vs. **I tried Bill to win*. Therefore, given the analysis so far, the system would erroneously allow a sentence such as *I tried for Bill to win*, just as if it were analyzing *want*.

This distinction *can* be captured if we are willing to use the k-tail method of induction. Suppose that either the Subset Principle or a principle of indirect negative evidence is invoked, so that unless positive evidence appears indicating otherwise, the system guesses as narrow a possible set of surface strings (a language) as possible. Alternatively, if negative indirect evidence is admitted, we may suppose that if the acquisition procedure does not see evidence of a certain construction P on a finite, simple set of data (say, sentences of degree 2 or less within a certain time limit), then it may conclude that construction P never appears in the language.[93] Either assumption is sufficient to make *I tried Bill to win* an ungrammatical string, without ever having seen it. But then, the lexical items *want* and *try* are known to be different, since identical strings lead to the accepting state for *want* and to the rejecting state for *try*: *I want Bill to win* is fine, but *I tried Bill to win* is not. This indicates that the equivalence class that held *want* and *try* together should be split. How should this difference be encoded? Minimally, we could do so by a diacritic feature, call it "F," that is +F just in case a verb is of the *want* type (and allows lexical NP's in embedded S complements along with *for* Complements)

---

93. For an analysis showing that the Subset Principle subsumes that of indirect negative evidence, see Chapter 3.

and -F if it is of the *try* type (and does not allow lexical NP's and *for* Complements). Note that some such features is required anyway, since there must be some way to distinguish between verbs demanding *for* Complements and those demanding *that*.

The correlation between lack of lexical Subject NP's and absence of *for* Complements is not accidental. Suppose that *for* is always present in forms such as, *I want Bill to win* -- at the reconstructed abstract level of S-structure. Note that there would be positive evidence for this abstraction in the form of sentences where *for* is not deleted, e.g., *I want very much for Bill to win.* Then the ±F feature would simply be a diacritic for whether or not a verb takes a *for* Complement. In the case of *want*, there would be evidence for the value +F; in the case of *try*, there would be no such evidence. Since the value +F allows for a wider range of surface forms, it would be most marked -- that is, the system would assume the value for a verb to be -F unless positive evidence was received to the contrary.

Finally, there is a class of verbs, including *seem*, *believe*, and *likely*, that behave differently from either *want* or *try* in that they do not allow lexical NP's as the Subject of their embedded Complements, but do allow *that*-Complements: *John believes that Sally is sick*, *John believes Sally to be sick*; *It is likely that John is sick*, *John is likely to be sick.*

How can the equivalence class of lexical items corresponding to this group of verbs be acquired? First of all, simple *that* complements will be handled as before: *I believe that John kissed Sally* will be parsed just like *I know that John kissed Sally.* Thus *believe* and *know* will be placed in the same lexical class.[94]

Now suppose that an example such as, *John believes Sally to be home* is encountered. There is no real difficulty in *parsing* this example, but how is it to be distinguished from, e.g., *I want Sally to be home*? Note that we cannot interpolate *that* as we did with *for*, since this would result in, *John believes that Sally to be home.* The key problem, of course, is that the NP *Sally* can appear in a tenseless clause only if it receives Case from an adjacent PP or Verb. In the usual case, the S clause boundary blocks this assignment.[95] One straightforward solution, then, would be to simply add a diacritic feature that indicates that Case assignment applies in this situation, where it ordinarily would not -- Exceptional Case Marking. Then *believe* and other verbs like it would be assigned this feature, as indicated by the positive examples, *John believes Sally to be ill*, *Sally seems to be ill*, and so forth; verbs that take *that* Complements but do not appear in such constructions, such as *probable* -- *It is probable that John will*

---

94. As we will discuss below, this means that the default assumption (unmarked) assumption is to exclude cases such as, *John believes Sally to be a fool*, that is, that S deletion is not permitted until evidence is received to indicate otherwise. This ordering of hypotheses follows the Subset Principle.

95. As indicated by the ungrammaticality of, e.g., *John tried Bill to win.*

win but not **John is probable to win* -- would not receive this feature. In short, this difference in positive example sentences would prompt a class split of Verbs. [96]

Let us summarize the discussion of Verb Complement structures.

1. A substantial portion of the complement structure of English Verb Phrases has been acquired. Reviewing the Marcus grammar, we see that Verb typology described there has been almost completely acquired, save for "small clause" constructions (e.g., *I like John drunk*). These constructions include *that* Complements, *for*-Complements, deleted *for* Complements, more complex NP-S subcategorizations (e.g., *persuade*, and exceptional cases with the matrix verb governing the Subject of the embedded clause (e.g., *believe*):

The Marcus Verb Complement System

1. If the verb is passive then activate passive and run passive rule;
2. If the verb can take an infinitive-object, then:
   2.1 If the verb can take an infinitive without *to*, then
   activate **to-less-infinitive**
   2.2 If the verb can take an infinitive without *to be*, then
   activate **to-be-less-infinitive**
   2.3 If the verb takes 2 objects, one an infinitive (e.g., *persuade*)
   then activate **2-object-infinitive-object**
   else activate **infinitive-complement;**
   2.4 If the verb takes an infinitive object without a subject (e.g.,
   want), then activate **subject-less-infinitive-complement**
   else if the verb takes a delta subject (e.g., *seem*)
   then activate **no-subject.**
3. If the verb takes a that-complement, then activate **that-comp**

Note that all categories save for 2.1 and 2.2 have been acquired. Passives were dealt with in the previous section. *To* infinitives with and without Subjects, encompassing three different classes of verbs -- the *try* type, the *want* type, and the *believe* type -- have been acquired. The second half of clause 2.3 and clause 2.4 of the Marcus typology above are meant to handle these cases. Verbs that take NP-S complements, e.g., *persuade*, have also been acquired; these fall under the first part of Marcus' clause 2.3. *That* complements have been acquired, and are handled by clause 3.0 of Marcus' typology.

---

96. This difference can be *represented* in many different ways. One way that has been suggested is via a rule of S deletion: since the reason Case assignment is blocked is the presence of the S boundary, verbs like *believe* could simply be assumed to delete or remove this phrase boundary. This is plainly a purely mechanistic solution that demands a better explanation.

Note that it is unlikely that the split between these verb classes could be made on a purely semantic basis, as has been pointed out in the literature: *probable* and *likely* are virtually synonymous.

2. The acquisition of S complements of Verbs is parasitic on the prior analysis of NP complements, in the sense that a "+Argument" requirement is first acquired by the analysis of sentences with simple NP Objects. The use of a "+Argument" feature is a familiar way of describing the alternation of NP and *S* complements.[97]

3. The use of an incremental acquisition procedure that "gives up" if a construction is too complex can be used to suggest (if not explain) certain developmental orderings. For instance, The analysis of embedded S's could well be easier that that of S's with full lexical NP's because of a reduced possibility of confusion with the NP Object case. This same line of argument also shows that Object embedded S's and relative clauses will have a tendency to appear before Subject embedded S's (e.g., infinitivals) or Subject relatives, as is apparently the case. In the context of the acquisition procedure, the reason for this is that Object embedded S's and relatives have a clear-cut S boundary, namely the end of sentence marker, that Subject S's do not have. The additional difficulty of establishing the boundary between the end of the Subject S and the root S means that these sentence types will be fixed only after the easier Object cases are successfully analyzed.

4. Several problems have been dealt with via description, but not really explained. These include the diacritic feature markings on Verb classes for the ability to take a *for* Complement, or the ability to have the matrix Verb Case mark the Subject of the embedded S. The current system can successfully acquire the right feature markings, but not in any principled way; it could just as easily acquire the reverse marking system.[98]

---

97. See Stowell [1981] for a more thorough analysis of this property.
98. Several topics concerning Verb complements have not been covered here.

One is the difference between so-called Subject control and Object control verbs has also not been covered here: *I promised Bill to leave* -- *I persuaded Bill to leave*. On any account, there must be some way to learn that in the first case, the Subject of the embedded clause is *I*, and in the second case, it is *Bill*. Since the two sentences form a minimal pair, there must be some other source of information that is drawn upon. The obvious solution is to rely on extra-syntactically reconstructed information about the meaning of the sentences.

Second, because all traces have been required to have antecedents, the system cannot handle cases where there is a phonologically empty NP that is "arbitrary" in reference, as in, *To go home would be foolish*. Therefore, these *to*-infinitives in Subject position cannot be acquired. An extension to include arbitrary reference traces (Pro) would not be difficult, however.

Finally, so-called "small clauses" have not been considered, constructions such as, *I think Bill a fool. John seems unhappy.* In Marcus' parser, these constructions were analyzed by inserting a *to be* into the input buffer, as appropriate.

2.6.4 Simple *Wh*-questions

Having covered the extension of the X system to include embedded sentences, we can turn now to *Wh*-questions. There are some simple cases to deal with first.

The "easiest" cases of *Wh*-constructions are those that already fall under the core Operator-Operand structure acquired in the analysis of S's. Consider for example,

Why did John kiss Sally?
Opr--Operand
WH------S

How did John kiss Sally?
Opr--Operand
WH-----S

If sentences such as, *I know why John kissed Sally* are encountered after sentences such as, *I know that John kissed Sally* have been successfully analyzed, then *know* will be established as taking a Propositional Complement. *Why* will be analyzed just as *that* is -- as an element in Operator position, the Head of the S. The trailing S can already be parsed by known grammar rules. Further details of this process will be omitted here.

Given this analysis, then one would expect that the very first *Wh* constructions would behave as if they were purely extensions to the X system, introduced via a rule something like,

S-bar -->WH S

(Note that this rule is presented purely for illustrative purposes; it need not be explicitly represented, since the Operator-Operand configuration already encodes it, with WH = Operator, and S = Operand.)

In fact, Labov and Labov note [1976] in their exhaustive analysis of their daughter's use of *wh*-questions that something like this seems to occur; the earliest *wh*-forms act as if they were introduced via the phrase structure rule listed above.

Now consider simple *wh* questions where there is a "gap" -- i.e., where the *wh* element corresponds to an item displaced from its canonical argument position:

Who will John kiss?
Who kissed John?

Let us suppose, as argued above, that the straightforward *wh*-forms have already been acquired, so that

wh is known to fill an Operator position.[99]

What happens then when *Who did Sally kiss* is to be analyzed? Assuming that sentences such as *Why did Sally kiss John* can already be parsed, then we may presumed that *who* is recognized as a *Wh* element, just like *why*. Note that again the trigger here is basically phonological, grounded on the similarity between *why* and *who*; in this respect, it is similar to the dual use of *for* as Preposition and Complementizer. In any case, the *wh* element will be attached and the trailing S analyzed, just as in the simpler *Why did Sally kiss John*. Crucially, the S node heading the entire tree will be annotated *+ Wh*, as usual, corresponding to the name of the grammar rule that attaches the *Wh* element to the S. When the parse reaches the analysis of the Complement of the Verb *kiss*, then, just as in the case of passives, analysis halts because the buffer now holds just the end of sentence marker # and there is no grammar rule that matches the current parser state.[100]

The acquisition procedure is entered. *Attach* and *switch* both fail, for the obvious reasons. *Drop trace* is then tried. It succeeds, since the Attach_NP grammar rule can execute after the *Drop trace* is performed. This new rule is saved; note that it requires the annotation *+ Wh* on the S node for it to trigger, as is appropriate.

The previous example was a simple one, because the missing NP Object was clearly signalled. Its Subject counterpart is not so easy to acquire, because the "gap" -- the missing Subject -- is not detectable on purely phonological grounds, at least not in any obvious way; the gap is string vacuous.[101]

Who kissed Sally?

In this case, the leading *wh* element will be parsed as before, attached as the COMP (the operator or Head) of the S that dominates the parse tree. Next, the Complement portion of this phrase is entered. The items now in the buffer are:

```
-----------------------------------
| kissed  +V -N +Arg|  Sally +N -V|  #  |
-----------------------------------
```

---

99. If *wh* is an operator, then it is not an argument, at least under the assumptions that we have made so far. There is some reason to believe that *wh* is indeed not an argument. See Chomsky [1981].
100. Recall that the passive rule demands the flag *+passive* for it to trigger, and this feature is not present here.
101. We ignore here stress patterns, pauses, or intonation, which may in fact provide some indication of a gap.

Depending on our analysis of INFL, a new X-max will be projected on the basis of *kissed*; let us assume that the *ed* morphology prompts the creation of an INFL-max (S). As usual, this causes the packet system NP--INFL--VP to be entered. First we activate rules to parse the Specifier of the INFL-max, NP. No rules match, but this NP is obligatory (for English). As a result, the acquisition procedure is entered. *Attach* fails because *kissed*, or rather the Inflection *ed*, is known to form the Head of the phrase. *Switch* fails because a *switch* violates the Operator-Operand structure *kiss-Sally*.[102] *Drop trace* succeeds; there is an antecedent element to bind to the NP-trace as well, the *wh* element.[103] The trace will be attached as the Subject NP by the known Subject_NP_Attach grammar rule, and the remainder of the sentence will proceed normally.[104]

To close this section, we will briefly discuss two constructions that fall under extensions of the S format, negative adverbial constructions and Topicalization.

Negative adverbials are forms such as, *Never have I seen such a mess!*. The Marcus parser did not have a grammar rule to handle such cases, but one is easily acquired. *Never* is presumably known as a [-N -Arg] item, an Operator taking a Verb operand, as in, *I have never seen such a mess.* If this analysis is correct, then *never* will be projected just like *why* or *how*. The remainder of the sentence will be parsed like a typical aux-inverted case.

Topicalizations are constructions such as, *Beans, I hate*. Plainly, the fronted NP is not an operator, like *why* or *who*. Therefore, attachment of the NP cannot proceed if the COMP element is a non-argument. One extension to handle this -- not currently implemented -- is to introduce an explicit TOPIC node as part of the expansion of S. This position might also be reserved for the Subject NP of passive sentences.

2.6.5 *There*-insertion

Sentences with existential *there* are not difficult to learn to parse, but the exact analysis is harder to come by.

---

102. Note that the system must appeal to its supplied predication representation of the sentence in this case.

103. What happens in languages that permit a missing Subject NP, or NP inversion, e.g., Italian? In this case, the expansion of S is something like (NP) INFL VP. As stated, the acquisition procedure will not successfully learn that the Subject NP is optional in such languages; it will try to drop a trace, and fail, because there is no antecedent. To modify the procedure, there could be a rule like that used in imperatives in English, that inserts a phonologically null element with reference to the speaker instead of the person adressed. This approach has not been implemented, and the problem will be simply left unsettled here.

104. Note that this is not quite accurate: in a sentence such as, *Who will kiss Sally*, the trace will be dropped immediately after *who*, and there will be no reversal of Subject-Aux inversion. Therefore, this sentence will not be on a par with *Who did kiss Sally*, not a desirable result. This problem too will have to be left unresolved here.

For our purposes here, the key point about *there* is that it behaves like an NP; it enters into agreement with its Verb and Object and appears in NP positions.

There was a riot on Tuesday.
*There were a riot on Tuesday.

How might this be established?

One w : this could come about is by the familiar process of phonological triggering, namely, the use of *there* as an NP in other contexts. Another use of *there* is demonstrative or presentational:

There is a lion. (Compare, *That* is a lion.)
A lion is over there.

Although these uses of *there* are different, from the standpoint of acquisition they are intimately related. Demonstrative *there* picks out a spatio-temporal location, and so has a semantic ground as a substantive thing. Thus it is natural to suppose that this *there* has the features [+N -V]. Then an example such as *The lion is over there* will prompt the creation of grammar rules to parse *there* as the Head of an NP. Given this analysis, a demonstrative *there* in Subject position would also be analyzed as an NP. How would agreement work? Nothing has been said so far about Subject-Verb agreement, but however that process works, in copulative sentences the Object forces agreement as well: *John is a nice guy* but not *John and Bill are a nice guy*. Thus one could simply use the features of the Object NP to force agreement, at least in these cases.

It is this analysis of demonstrative *there* that is assumed to pave the way for existential *there*. Since the acquisition procedure does not really know about the *semantic* interpretation of the structures it builds, it will simply plunge ahead and parse *There was a riot* just like the demonstrative counterpart. Agreement will be required because of the copulative, enforced by whatever mechanism is used in other cases of copulatives. (Perhaps this mechanism is something like the device of indexing, but we remain neutral on the details.)[105]

In short then, the analysis of *there* sentences is another case where it seems likely that the acquisition of semantically clear examples precedes the analysis of grammaticalized, more syntactically-based constructions of the same type.

The next two subsections discuss simple extensions to the acquisition procedure that have been designed but not implemented.

---

105. Cases of apparent right-ward movement -- extraposition and the like -- are not covered here. Thus we do not cover the cases, *There was believed to be a riot* or *It was likely...*

2.6.6 Imperatives

The acquisition procedure also acquires a grammar rule to deal with imperatives, given an extension to its list of grammar rule actions.

The gist of imperative constructions is that a designated lexical item, *you* is evidently inserted as the Subject NP of a tenseless Sentence:

> Be quiet -- You be quiet.
> Go home -- You go home.

Tag questions show that it is indeed *you* that is inserted:

> Be quiet, won't you?

Note that tenseless sentences ordinarily do not allow lexical NP's to surface phonologically; recall that it is the tense element that assigns Case in root sentences to the NP and permits it to be pronounced:

> John is quiet -- *John be quiet -- I want (for) John to be quiet.

This raises a certain problem for the acquisition procedure. Its rules must be designed to reject tenseless main sentences. That is, it must *not* in general be able to parse a sentence such as, *You be good.*[106] Since this constraint is apparently applied at a level of phonological representation -- it determines whether an NP can be pronounced or not -- then it is reasonable to suppose that a designated lexical item inserted *after* phonological analysis will not be subject to the same constraint. That is, since *you* is apparently inserted into the input buffer after phonological processing, it is not constrained to appear in a tensed (root) clause, as it would be if it were a phonological word.

To carry out this analysis we must make some changes to the acquisition procedure that have not been implemented as yet. First, we must add an action that attempts to insert one of a designated number of lexical items into the first position of the input buffer. What should these elements be? Plainly, *you*

---

106. This may not be quite right, since this is a common dialect variation: *You be good while I'm gone, now!*.

Perhaps this is handled via an implicit modal: *You be good* is actually *You will be good* -- as indicated by the tag question, *Be good, won't you?*

Note that Marcus' grammar as described in Marcus [1980] has this problem. It blithely parses tenseless root sentences, e.g., *I take a book* or *I be happy*, since it is designed to parse *Take a book* by inserting *you* and then proceeding just as if the sentence were a simple declarative.

The conclusion is that there must be some way to encode a difference between a parse that inserts a *you* into the buffer and one that has *you* or another lexical NP presented to it via ordinary means.

is one such item. Another potential candidate is *to be*, as in *I want John kissed--> I want John to be kissed.* In any case, the list is finite and small.[107] The test for the success of this action should be like that for *switch* or *drop trace*: if a known grammar rule executes after the *insert <item>*, then the action has succeeded; otherwise, it has failed. Where should it go in the ordered list of attempted actions? According to the Subset Principle, we want to order hypotheses so that the narrowest possible class of languages is generated. As before, it is more conservative to place it after one has attempted to analyze the construction as a base form -- that is, to place the action after an attempted *attach*. With respect to *switch* and *insert <item>*, evidently *switch* fails on a case such as *Take a hike.* The NP *a hike* will be moved into Subject position, but no known grammar rule will match afterwards, since the pattern to attach Subject NP's demands a buffer pattern with either a +V -Arg item in the second buffer cell and -N in the third position (*I walked. I walked to school*, or else a +V +Arg item in the second cell and a +N item in the third (*I ate a candy*). So the order here will not matter. Finally, since *drop trace* allows for a much broader range of surface forms, it should be ordered last. Thus we obtain the ordering for attempted actions of: (1) attach; (2) switch or insert <item>; and (3) drop trace.

2.6.7 Dativization

So-called dative-movement rules, e.g., *I gave a book to John -- I gave John a book*, are discussed in detail in Chapter 3, Section 3.

2.6.8 Diagnostic Rules

In Marcus' grammar, diagnostic rules were those used to distinguish between minimal pairs such as:

> Have the boys take the exam.
> Have the boys taken the exam?

For the most part, the job of these rules is to decide between alternative categorizations of lexical items -- in the case above, between *have* as a main verb and *have* as an auxiliary verb.[108]

Can diagnostic rules be acquired? At least in this simple case, the notion of "diagnostic rule" can be replaced by invoking the ordering principle presented earlier that specific rules should execute before general rules. Suppose that the sentence, *Have the students taken the exam?* is encountered. As usual,

---

107. Lasnik and Kupin [1977] also include this action in their list of primitive transformational operations.

108. Diagnostic rules could also be used to determine alternative constituent attachments. Note that the determination of category labelling and constituent attachment *exhausts* what must be done for correct parsing, so that the notion of a "diagnostic rule" is not a very precise one.

have will trigger the creation of an X-max item, with features +V, -N, and (in this case) both +Tense and -Tense corresponding to its dual use as either a main verb or an auxiliary verb. Thus the machine state will have a current active node of X-max [+V -N -tense +tense], the system will be parsing the Specifier of this X-max, and the buffer will hold the items *Have*, *the*, and *boys*, respectively. Assume that the NP *the boys* is parsed via a subroutine call to the parser (an attention shift), as we have assumed in other examples. Then the buffer now holds the following items:

```
-------------------------------------------------
| have +V -N +tense -tense| NP | taken +V -N +tense |
-------------------------------------------------
```

Now, two rules match this buffer: both the imperative rule and the switch rule. However, the buffer pattern for imperative is more general than that for switch:

```
Buffer pattern for imperative (with transitive verb):

    ---------------------------------
    |  +V -N -tense +Arg|  NP  |  *  |
    ---------------------------------

Buffer pattern for switch:

    -------------------------------------
    |  +V -N +tense |  NP  |  +V +tense  |
    -------------------------------------
```

The second pattern is more specific, since it demands that some predicates be true of the third buffer cell, where the first does not.[109]

Therefore, by the protocol (derived from the Subset Principle) that specific rules should execute before general rules, the switch rule will get first crack at this example. In this case, it succeeds, and its effect is to block the imperative rule. (Note that the *+tense* feature will now be percolated through to the X-max, eliminating the *-tense* marker from now on.) If the example is one with a tenseless verb, e.g., *Have the boys take...*, then the switch rule will again be attempted first, but will fail. Now the system will attempt to match the imperative rule, and succeed, inserting *you* as required. Thus, in this

---

109. One could perhaps also use simply the number of predicates as a measure of complexity, and brake ties randomly.

case a principle derived on grounds of learnability -- the execution of specific before general rules -- also solves a problem of rule ordering in parsing.

2.7 Summary of the Chapter

Chapter 2 has covered much ground, and it is worthwhile reviewing what has been accomplished before proceding to the formal analysis of acquisition, the topic of Chapter 3.

The main aim of this chapter has been to describe an acquisition procedure based on an on-line parsing model. The first sections of this chapter discussed the the assumptions behind the model, and the changes made to Marcus' original parsing procedure in order to make it more amenable as an acquisition model. These changes include: (1) the elimination of explicit rules for labelling nodes, using a "percolation" principle; (2) the removal of explicit packet activation and deactivation rules and their replacement by an automatic instantiation of X schemas; (3) the use of complex feature bundles to label non-terminal nodes, rather than standard category labels such as VP, NP, and the like; (4) the elimination of all actions except three (four, if the extension to *insert* ⟨*item*⟩ is admitted).

The next two sections discussed the acquisition of base phrase structure rules and the acquisition of grammar rules to handle displaced constituents. The base rules acquired can be viewed as lexical insertion contexts, as introduced by Chomsky [1965] and described formally by Peters and Ritchie [1969] and Joshi and Levy [1977b]. The knowledge so acquired is in the form of lexical equivalence classes, items that "behave alike" with respect to parsing contexts. No *explicit* system of context-free re-write rules is acquired. Lexical category ambiguities, it should be noted, have a natural place within this framework; several examples were presented to show that early on a lexical item may be almost completely unknown in its features, and yet can be analyzed in a "top-down" fashion so as to establish its identity.

The movement-type rules acquired include rules to handle Auxiliary inversion, passives, missing Subjects in embedded clauses, and simple *wh*-movements. Many constructions have *not* been acquired. Some demand simple extensions to the current acquisition procedure; these have been hand simulated, and include imperatives and one or two diagnostic rules. Other constructions cannot be acquired because they require extensions to the representational system assumed so far. For example, there is no means to distinguish between trace or PRO, or, to put it another way, between empty NP's that are governed and those that are not. Only governed empty elements have been allowed. Therefore, Subjectless *to*-infinitives in Subject position are not analyzable. Besides this case, right-ward movement constructions generally have not discussed because no means has been presented to bind traces that appear to the left of their antecedents. As a result, right-ward extraposition has not been considered. Despite these gaps, a substantial portion of Marcus' grammar

has been acquired automatically (See the Appendix to this chapter for details).

Besides the plain facts of what knowledge can or cannot be acquired, the acquisition procedure is broadly compatible with the course of acquisition that one might expect. Base rules and the major Head-Complement order is fixed first, drawing upon extra-syntactic knowledge as a cue. Once this basic constituent order is fixed, the acquisition procedure builds on what it knows to "grammaticalize" forms so that a reliance on extra-syntactic knowledge may be dropped. In all of this, the Projection Principle plays a major role, ensuring that semantic transparency holds. Finally, the incremental character of the acquisition procedure seems to lead to the right developmental predictions in many cases: the order of acquisition of base rules (before movement rules), passives, embedded S's without Subjects, and Object clauses before Subject clauses follows from the way that the acquisition procedure gives up if an example is too complex. This same incrementality is also crucial in ensuring that the acquisition procedure bases its guesses on clear cases, helping it to avoid over-generalizations from which no recovery is possible.

Appendix I - A Listing of Rules Acquired

Grammar Rules in Parsifal and Grammar Rules Acquired

Total # of Rules in Parsifal[110] = 107, of which 83 are purely "syntactic" rules.
Total # of Rules that can be currently acquired = 58, about half of all the rules, or three-quarters of the syntactic rules.

110. All rules dealing with number expressions, clock times, and the like, hand-tailored for the personal assistant project, are not included in this count.

Listing of Rules Acquired

The form of the rules acquired is as follows:

Packet name

Pattern: features of
{current cyclic node} (including packet name and annotations)
{current active node} (including packet name and annotations)
{buffer state}

Action: attach/switch/drop trace/insert <lexical item>

where *Packet*= the name of the context-free base component currently active when the rule was acquired, e.g., Parse-subject-Noun-Phrase, Parse-Verb-Phrase;

Pattern= three parts, (1) the features of the current cyclic node, that is, the S(entence) or Noun Phrase node that immediately dominates the current active node; (2) the features of the current active node, that is, the phrase at the top of the active node stack, currently under construction; and (3) the features of items in the first three cells of the input buffer (either terminal items like single words or completely built constituents, like whole Noun Phrases). (The feature "*" in a buffer cell is a "wild card" indicating that *any* buffer item will match that rule pattern.)

Actions= *Attach* the item in the first cell of the buffer to the current active node; *switch* the contents of the first and second cells of the buffer; *drop a trace* (a dummy Noun Phrase node) into the first cell of the buffer; *insert a lexical item* into the first cell of the buffer, where a *lexical item* is one of {*You, be, to be, for, of*}.

Rules by Type

Transformational Rules: 10 Acquired: 8

1. Passive (*A meeting was scheduled for Tuesday.*)
2. Auxiliary verb inversion (*Was a meeting scheduled for Tuesday?*)
3. *there*-insertion (*There was a meeting scheduled for Tuesday.*)
4. (Simple) *Wh*-gaps (*Who scheduled a meeting for Tuesday?*)
5. Imperative (*Schedule a meeting for Tuesday.*) (Note: hand simulated only)

6-8. *Pro* insertion in embedded clauses (3 rules)

(*I promised to schedule a meeting.*)
9-10. *To, be* insertion in embedded clauses (2 rules)
(*I want a meeting scheduled for Tuesday.*)

Acquired:
All of the above except for the rules that deal with small clauses.

Phrase Structure Rules: 34 Acquired: 32

Main Sentence Phrases - 4 rules
Auxiliary Verb Phrases - 10 rules
Verbs - 4 rules
Prepositional Phrases - 3 rules
Verb Complement Clauses:
That-complements (*I knew that a meeting would be scheduled.*) - 3 rules
To-infinitives (*I want to schedule a meeting.*) - 2 rules
Object Phrases - 3 rules
Embedded Sentential Clauses - 2 rules
Labelling rules - 3 rules

Acquired: All of the above rules, except 1 of the *that*-complement rules, and 1 of the Verb rules.

Noun Phrase Rules: 24 Acquired: 11

Start Noun phrase - 2
Determiner - 1
Quantifier - 4
Adjective - 1
Noun - 1
Pronoun - 1
Proper name - 3
Other modifier - 3
Incomplete NP - 8

Two of the quantifier rules, the three "other modifier" rules, and the incomplete NP rules cannot be acquired.

Wh-phrase Rules: 7 Acquired: 4

Create *wh* phrases - 2

Reduced relative phrases - 1
Wh-PP phrases - 2
Finish *wh* phrases - 2

The reduced relative and *wh*-PP rules cannot currently be acquired.

Lexical Diagnostic Rules: 8 Acquired: 3
1. *Have* diagnostic (*Have the boys take/have the boys taken*)
2. *A* diagnostic (*A hundred pound bags/a hundred pound bag*)
3. *Which* diagnostic
4-6. *That* diagnostic (*I know that boy with red hair/I know that John has red hair.*) - 3
7. *To* diagnostic
8. *What* diagnostic

The *have*, *a*, and one *that* diagnostic can be acquired.
The following rules require "goodness of fit" evaluations provided by the case frame interpreter, and, as such, cannot be acquired by the current system.

Wh "semantic" rules - 6
Other "case frame attachment rules" - 9
Prepositional Phrase attachments - 8
Other case frame - 1

The following briefly describes some other acquired base rules not covered in the main text.

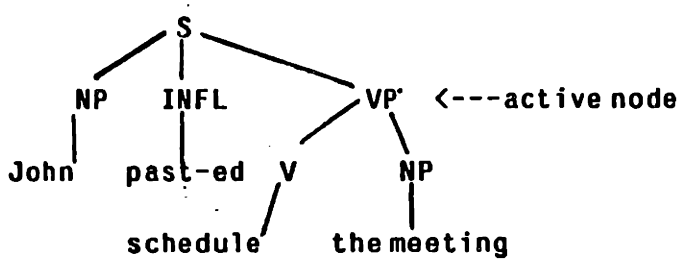
Prepositional phrase attachment.

Packet: Parse-Complement-X [+V -N +Arg]
Pattern:
Cyclic node: INFL-max (S)
Active node: X-max [+V -N]
Buffer: [PP][*][*]
Action: Attach

The effect of this rule is to attach a completed Prepositional Phrase to the Verb Phrase of a sentence. (A similar rule attaches PPs to Noun Phrases, e.g., *The man with blond hair.*)

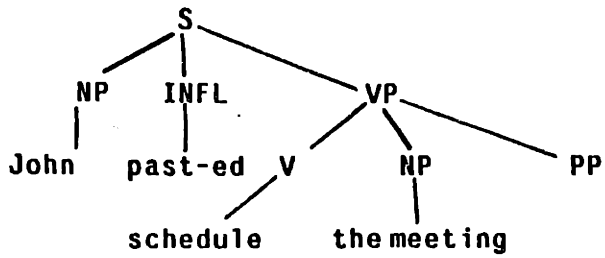
Typical sentences handled: John scheduled the meeting for Tuesday.

Parse tree before the rule:



Input buffer: |PP for Tuesday| # | empty |

Parse tree after the rule:



`Input buffer: | # | empty | empty |`

This rule does *not* deal with a host of complications concerning the alternative attachment of Prepositional Phrases, e.g., classic ambiguities in attachment such as, *I saw the man in the park with the telescope.* The resolution of such ambiguities apparently involves interaction with a "semantic" component, i.e., a device that can provide a preference ranking for alternative attachments. This acquisition of this preference component is not the subject of this research.

Rules to build Noun Phrases:

Determiner attachment.

Packet: Parse-Specifier-X [+N -V]
Pattern:
Cyclic node: INFL-max (S)
Active node: X-max [+N -V]
Buffer: [+N -V -][+N][*]
Action: Attach

This rule attaches a determiner like *the* or *a* as the specifier of a Noun Phrase.

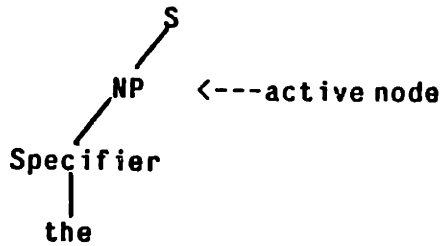
Typical sentences handled: The meeting, a meeting; one meeting.

Parse tree before the rule:



Input buffer: | the +N -Head | meeting +N -V +Head | was +V |

Parse tree after the rule:



Input buffer: | meeting +N | was +V | scheduled +V |

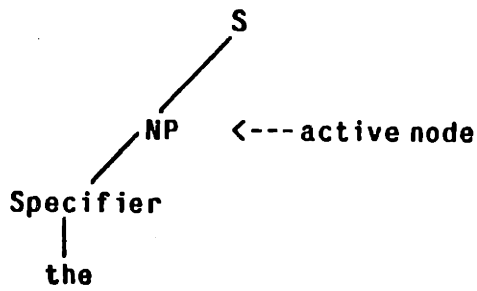
Noun attachment.

Packet: Parse-Head-X [+N -V]
Pattern:
Cyclic node: INFL-max (S)
Active node: X-max [+N -V]
Buffer: [+N -V +Head][*][*]
Action: Attach

This rule attaches a (head) Noun to a Noun Phrase under construction.

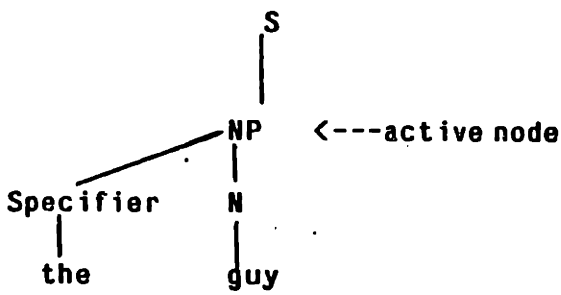
Typical examples: The guy scheduled a meeting; a meeting was scheduled.

Parse tree before the rule:



Input buffer: [guy +N +Head][schedule +V][a +N -Head]

Parse tree after the rule:



Input buffer: [schedule +V][a +N -Head][meeting +N +Head]

3. A Theory of Acquisition Complexity

3.1 Introduction: Simplicity, Acquisition, and Evaluation Metrics

From the earliest days of the study of generative grammar there has been an attempt to formulate explicit criteria against which alternative grammars are to be judged:

> A grammar of a language must meet two distinct kinds of criteria of adequacy. On the one hand it must correctly describe the 'structure' of the language...On the other hand it must meet requirements of adequacy imposed by its special purposes...or, in the case of a linguistic grammar having no such special purposes, requirements of simplicity, economy, compactness, etc.
> [Chomsky, *Morphophonemics of Modern Hebrew*, 1951, p. 1]

What is the import of the term "simplicity" as it used in this passage, or more generally in linguistic theory? For the most part, it was recognized that the notion of simplicity serves in the same role as it does in the other natural sciences, namely, as a criterion meant to highlight law-like generalizations while supressing irrelevant detail. A simplicity criterion thus goes hand in hand with a particular notational or representational system, a language for writing down theories of grammar. If the notational system adequately captures the "linguistically significant generalizations" of a language -- an empirical issue -- then this success ought to be verified through a correspondence between the laws capable of simple notation in that system and the observable regularities of the language.

As an example, consider the following hypothetical set of phonological rules as described in *The Sound Pattern of English*, p. 333:

i--> y/___p	i--> y/___p
i--> y/___r	i--> l/___r
i--> y/___y	i--> p/___y
i--> y/___a	i--> n/___a

> (As is customary, the notation "i-->y/___p" is to be read as "*i* is re-written as *y* if *p* is immediately to its right".)

The first set of rules, but not the second, plainly contains a regularity that demands attention: *i* is mapped to *y* in an entire set of environments. In contrast, the second set of rules contains no such regularity. This distinction can be reflected in the notational system adopted in SPE since one can use the notational device of braces (denoting disjunction) to write the first set of rules more compactly than the second:

$$i \rightarrow y / \_\_ \left\{ \begin{matrix} p \\ i \\ y \\ a \end{matrix} \right\}$$

Note that if one attempts to write down the second set of rules using braces then one winds up with an expression that is just as long as before.

So one part of what "simplicity" means hinges on the notational devices available in a language for theories that enables us to use length or compactness as a proxy for theoretical generalizations. This approach also dates from the earliest days of generative grammar:

> We want the notion of simplicity to be broad enough to comprehend all those aspects of simplicity of grammar which enter into consideration when linguistic elements are set up. Thus we want the reduction of the number of elements and statements, any generalizations, and, to generalize the notion of generalization itself, any similarity in the form of non-identical statements, to increase the total simplicity of the grammar. As a first approximation to the notion of simplicity, we will here consider the shortness of grammar as a measure of simplicity, and will use use such notations as will permit similar statements to be coalesced...The criteria of simplicity governing the the ordering of statements is as follows: that the shorter grammar is the simpler, and that among equally short grammars, the simplest is that in which the average length of derivation of sentences is least.
> [*Morphophonemics of Modern Hebrew*, pp. 5-6]

Choice of syntactic connectives such as braces, brackets, and the like, along with a particular set of rules for expanding an expression written using those connectives thus plays a role in what is judged simple or not simple. But the measurement of simplicity does not end there. More generally, like any representational system a notational scheme (or language) consists of two basic parts: a set of vocabulary symbols (such as NP, VP; or *p*, *a*), and a set of connectives (such as { }, /, [ ]) along with a set of rules for expanding expressions in the notational language. As we have seen, the choice of connectives matters, in that we can express generalizations using braces that we otherwise might not be able to express (assuming all other aspects of the representational language are held constant). The choice of basic vocabulary symbols is also crucial, however, as Chomsky and Halle have demonstrated. For example, we want the following set of phonological rules to be representable as a generalization (=have a short representation)

$$a \rightarrow ae / \_\_ \left\{ \begin{matrix} i \\ e \\ ae \end{matrix} \right\}$$

The reason is that the environment in which the change occurs consists of just front vowels -- intuitively, a "natural class" of vowels. In contrast, the following set of rules seems unrelated in their triggering environments:

$$a \rightarrow ae / \_\_ \left\{ \begin{matrix} p \\ i \\ z \end{matrix} \right\}$$

To represent the first set of rules more compactly than the second demands not a change in syntactic connectives, but a change in what is considered a unit of the notational vocabulary. In this case, Chomsky and Halle have argued that the "true" units of analysis are not the phonemes *a, p, z*, but rather bundles of *distinctive features* into which each phoneme is decomposed. Given this approach the grouping *a, e, ae* can be shown to have the distinctive feature values [+vocalic, +front] in common, hence can be completely described by a small quantity of information, whereas the grouping *p, i, z* has no feature values in common, hence can only be described by individually listing the features of *p, i,* and *z*. This list is enormously longer than the two features [+vocalic +front]. Hence the first set of rules, but not the second, expresses a generalization of the environment in which the a-->ae rule operates. Below and in Chapter Four we shall show that this intuitive distinction in length actually completely characterizes, in an information-theoretic sense, the distinction of an *explanation* vs. a *stipulation.*

To preview those results here, let us say that a description of a set of surface data expresses a *regularity* or constitutes an *explanation* if the description of the data is shorter than a simple list of the data itself. Thus the distinctive feature analysis expresses a regularity in the first set of a-->ae rules, because just two features (two bits of information) are sufficient to characterize the class of front vowels, whereas if we consider there to be 7 front vowels, then an outright list of the front vowels would take at least 7 bits of information to store in a table. One can see then that the distinctive feature characterization of the front vowels "compresses" the description of the front vowels. Succinct representations, then, are the hallmark of explanations. Compare in this regard Halle's definition of a "natural kind":

> *N* is a natural kind if fewer features are required to designate the class *N* than to designate any individual sound in *N*.
> [1961, p.90]

Chomsky's informal characterization is nearly the same:

> We have a generalization when a set of rules about distinct items can be replaced by a single rule (or, more generally, partially identical rules) about the whole set, or when it can be shown that a "natural class" of items undergoes a certain process or set of similar processes.
> [1965, p. 42]

As shall become apparent in the formal analysis below, the intuition that a generalization exists when a set can be replaced by a rule in fact exhausts the notion of generalization and defines what is or is not a natural class. In sum, what lies behind a generalization or a regularity in a set of data is a pattern that permits a "compression" in the description of that set, beyond a simple list of its members.

Plainly then, both vocabulary, connectives, and expansion rules are necessary in order to evaluate a notational proposal. Together they act as a kind of programming language for "writing down"

sequences of strings, be they strings of phonemes, syntactic categories, or units of semantic representations, the language in which a theory of grammar is written.

The advantages of this formal approach are several. For one thing, it allows us to settle certain questions about the role of notational systems in expressing linguistic generalizations, as raised by Chomsky and others in the early study of generative grammar. As an example, consider the problem of whether the choice of notation matters in the ranking of alternative theories, a problem raised initially in MPMH. In one sense it clearly does; witness the first pair of examples above, where the brace notation permitted a considerable economy of notation.[1] This example shows that if a notational system is not sufficiently rich then there may be some regularity that is inexpressible in that system, but easily captured in another.

This observation immediately leads to the question of whether one can define more precisely the term "sufficiently rich". Note that this question makes no sense without a formal framework in which the expressive power of a theory can be calculated. The framework chosen here will be drawn from that of automata theory, in which the expressive power of a theory is measured by the class of languages that the grammars specifiable by the theory can generate. For example, in this framework a theory that can specify grammars for every recursively enumerable set is greater in expressive power than a theory that restricts its grammars to consist of re-write rules that are just left-linear or right-linear (and hence can generate only regular languages). Other hierarchical distinctions, e.g., in terms of machine classes or time and space bounds, are also possible under this account of expressive power. As will be demonstrated in Chapter 4, shifts to more powerful notational systems can capture generalizations more successfully than weaker notational systems in two different situations. First, it is obvious that a weaker system may be unable to even *describe* certain sets -- as is the case if a notation that uses context-free grammars attempts to describe a strictly context-sensitive language. Second, and less obviously, it may also be true that while a weaker notational system can describe a set, it cannot do so compactly, but only via what amounts to a list of the elements of that set. In this case the weaker notation is inherently incapable of capturing descriptive generalizations about the set. In Chapter 4 it is shown that such cases arise in the study of grammars for natural language.

Given this programming system approach, then the answer to Chomsky's question regarding the effects of notational changes can be expressed more precisely by replacing the term "sufficiently rich" with "possesses a universal partial recursive function," that is, a "program" that can simulate all other partial recursive functions, hence all other notational systems, assuming that notational systems are

---

1. But see below in this chapter, where it is suggested that the brace notation in particular is dispensable if one is permitted to re-formulate vocabulary categories appropriately.

identified with the partial recursive functions. Let us call such a system a *universal notation.*[2] Let us now reconsider the possible effect of altering one's notational system. Consider the following example discussed in *Aspects of the Theory of Syntax*, p.43, regarding the formulation of a "compact" set of rules for the English Auxiliary system:[3]

> (15)Aux--> Tense (Modal)(*Perfect*(*Progressive*)
> Rule (15) is an abbreviation for eight rules that analyze the element Aux into its eight possible forms. Stated in full, these eight rules would involve twenty symbols, whereas rule (15) involves four (not counting Aux, in both cases). The parenthesis notation, in this case, has the following meaning. It asserts that the difference between four and twenty symbols is a measure of the degree of linguistically significant generalization achieved in a language that has the forms given in list (16), for the Auxiliary Phrase, as compared with a language that has, for example, the forms given in list (17) as the representatives of this category:
> (16)Tense, Tense--Modal, Tense--*Perfect*, Tense--*Progressive*,
> Tense--Modal--*Perfect*, Tense--Modal--*Progressive*, Tense--
> *Perfect*--*Progressive*, Tense--Modal--*Perfect*--
> *Progressive*
>
> (17)Tense--Modal--*Perfect*--*Progressive*, Modal--*Perfect*--
> *Progressive*--Tense, *Perfect*--*Progressive*--Tense--Modal,
> *Progressive*--Tense--Modal--*Perfect*, Tense--*Perfect*,
> Modal--*Progressive*
>
> In the case of both list (16) and list (17), twenty symbols are involved. List (16) abbreviates to rule (15) by the notational convention; list (17) cannot be abbreviated by this convention.

Chomsky observes that the set described in (17) *can* be abbreviated if a different set of notational conventions is admitted, namely, some notion of cyclic permutation. For example, one could re-interpret parentheses surrounding a list of elements as denoting the set of cyclic permutations of those elements -- a standard mathematical notation, in fact:

> (Tense Modal *Perfect Progressive*)

But a universal notation can simulate an arbitrary program p written in any other notation A_i via the program $p_u = 0^i1p$, where i is the index of the notational system A_i in some enumeration of all notational systems (partial recursive functions). In essense, the universal notation simply looks up the

---

2. More precisely, given an enumeration of the partial recursive functions, Φ_0, Φ_1,..., denote a universal partial recursive function by $\Phi_{univ}(i,x) = \Phi_i(x)$.
3. See also Chapter 3, section ? below for an analysis of the acquisition complexity of the English auxiliary system.

index of notation Λ_i and then "runs" p on that notation. But then, the size of p_u is just some constant (depending on the numeration of prfs) larger than p. Thus, given the alternative permutation notation, (17) is described by a program of roughly the same size as the program for (16) in the usual notation. Symbolically, $|p_2|$ wrt $\Lambda_j = |p_1|$ wrt Λ_i. In short, given a universal notational system, we have that $|p_u(\text{data set 17})| \leq |p_2|+(j+1)$, and $|p_u(\text{data set 16})| \leq |p_1|+(i+1) = |p_2|+(i+1)$. With respect to the universal notational system, compact representations of (16) and (17) differ only with regard to whether the program for notational system i appears before that for system j. But this enumeration order is arbitrary -- we could just as well have picked an enumeration of prfs that lists j before (i). In this sense, a sufficiently powerful notational system renders the demand to fix notational machinery moot.

This result would seem to indicate that a size complexity measure is uninteresting for "complete" computational systems. However, this invariance is *not* necessarily preserved by shifts to "weaker" notations, e.g., those that can specify grammars only for context-sensitive languages, or even only just some restricted subset of all the recursively enumerable languages (as is the case with modern transformational theories that can only specify a finite number of "programs," even if these generate languages that are perhaps not context-free or context-sensitive). Presumably this is the situation that holds with regard to linguistic theories; a desideratum of current theories of grammar is that they should be tightly constrained, specifying only a finite number of grammars, in some cases. In short, current theories of grammar are *not* universal programming systems; hence the results suggesting that program size measures are useless do not apply.

A second advantage of the formal analysis is that it may be used to study the problem of acquisition and to justify several of the "operating principles" used by the acquisition procedure discussed in Chapter 2. Informally, it has long been suggested that the notion of simplicity is intimately connected to acquisition, in the sense that what is simple, as expressed with respect to a particular notational system, is also easy to acquire. Children are assumed to search for regularities in the language of their caretakers, and are assumed to be predisposed to uncover certain regularities rather than others. But as we have seen, what is patterned is expressible via a short program; hence the linguist's criterion that short rule systems are desirable can be interpreted as a proposal about acquisition, rather than just esthetic sensibility. This is for example the suggestion made by Chomsky in the conclusion of the passage cited above:

> Hence, adoption of the familiar notational conventions involving the use of parentheses amounts to a claim that there is a linguistically significant generalization underlying the set of forms in list (16) but not the set of forms in list (17). It amounts to the empirical hypothesis that regularities of the type exemplified in (16) are those found in natural languages, and are of the type that children learning a language will expect; whereas cyclic regularities of the type exemplified in (17), though perfectly genuine, abstractly, are not characteristic of natural language, are not of the type for which children will intuitively

> search in language materials, and are much more difficult for the language-learner to construct on the basis of scattered data or to use. What is claimed, then, is that when given scattered examples from (16), the language learner will construct the rule (15) generating the full set with their semantic interpretations, whereas when given scattered example that could be subsumed under a cyclic rule, he will not incorporate this "generalization" in his grammar.
> [1965, pp. 43-44]

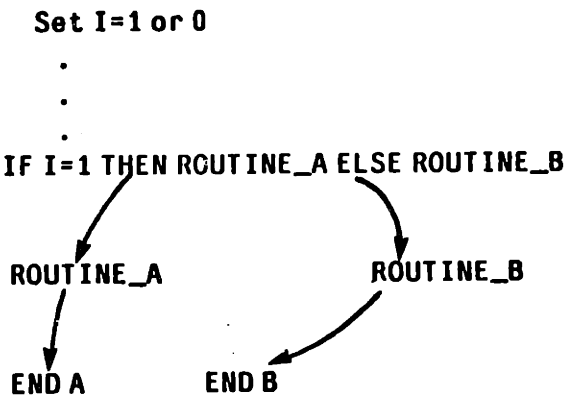
On this interpretation, simplicity means *acquisition simplicity.* To choose "short" grammars is to choose a grammar that is presumably easy to acquire. But what then of other possible functional demands on the language faculty? It has often been suggested that besides the demand of learnability, the demands imposed by language use -- the fact that sentences are spoken and understood rapidly -- must have played a role in the design of grammars.[4] The demand of processability is typically translated into a complexity measure by keeping track of the units of space or time used by a procedure that recognizes sentences generated by a grammar -- the familiar time and space hierarchies of automata theory. There are, then, at least two distinct notions of simplicity: (1) simplicity in the sense that it has usually been used in the linguistic literature (length of grammar, number of symbols) and (2) simplicity in the sense of the complexity of the resulting family of languages, in terms of time and space bounds demanded for recognition. These two measures correspond in an obvious way to the two most-often cited "functional demands" that one might imagine to be imposed on the design of the "language faculty" -- the ability to acquire language -- corresponding to the length of a grammar -- and the ability to use language -- corresponding to length of derivations in a grammar. A detailed examination of the trade-off between these two complexity measures, acquisition complexity and derivation complexity, is taken up in the next chapter. One remark, though, is worth making here. Recall from the material cited above that the earliest work in generative grammar (MPMH) adopted as a "tentative" measure of simplicity the length of a grammar, and then, within all grammars of equal length, length of derivation. Though tentative, formal re-analysis of this early proposal shows that it is right on the mark, or nearly so. Length of derivation can be shown (Book [1970]) to correspond to the time hierarchies used to classify grammars according to their weak generative capacity, and thus does serve as a proxy for recognition complexity. Similarly, grammar length serves as a proxy for acquisition complexity, as will be demonstrated in this chapter. Thus the proposal in MPMH in effect lays out the following research strategy for the study of language: find all grammars adequate with respect to a criterion of acquisition complexity; then further narrow this set by applying a criterion of derivation (recognition or production) complexity. What is interesting is that this approach still seems to be an appropriate one even after 30 years of additional research, in that the constraint of acquisition appears more potent in constraining the class of possible human grammars than the constraint of recognition. (For additional discussion, see Berwick and Weinberg, 1982).

---

4. See, e.g. Yngve [1960]; Marcus [1980]; J.D. Fodor [1978][1980]

One of the major aims of this chapter is to develop and apply a formal characterization of simplicity by pursuing the idea of a notational system as a programming language. The basic approach is to adopt the theory of *program size complexity* as the right yardstick for simplicity. This model meets several needs. First, it serves as a formal theory of *markedness*, by equating markedness as the term is usually used in linguistic theory as "the amount of external information required to fix a grammar." Second, this theory provides a developmental model of language acquisition (or, rather, grammar identification or instantiation), identifying the evidence that is required for acquisition with a sequence of program inputs, and the complexity of acquisition with the information-theoretic content of those inputs. Since an acquisition "program" consists of a sequence of computational states, starting in some initial state as specified by the theory of universal grammar and ending in a final state that is to correspond to the identification of a "correct" grammar, such a model is able to describe alternative developmental pathways, a matter of some interest to the psychologist. Moreover, since these developmental sequences (corresponding to possible computational sequences) are constrained to occur in certain valid combinations (corresponding to the possible and impossible computation sequences of a machine), it follows that an external observer tracking the successive "states" of such a procedure would observe it passing through only certain well-defined sequences and not others.

In particular, states can be described by a cluster of properties according to the implicational structure of the underlying program. Typically, setting the value of just one variable in a program can give rise to a number of immediate changes, some perhaps quite distinct. For example, the call to one or another of two completely distinct subroutines can be triggered by the value of a single other variable:



In this case, the "parametric" structure of the program is revealed by observing characteristic "snapshots" of variable values as the program executes. If I=1, then Routine A is selected, and since A is assumed to be distinct from Routine B, the values of variables set while executing the block of

code associated with A will be different from those associated with routine B. These logical dependencies between variable values can be described by saying that the execution of the statement I = 1 implies an entire block of subsequent variable value changes, namely, those occuring in Routine A. It is this *clustering* of variable value modifications that induces a second-order structure to the program, above the level of individual lines of code. Wherever Routine A is called, all its statements are executed together as a group. The "common fate" behavior of Routine A's variables is what makes Routine A identifiable as a functional unit -- a block of code that can be labelled as a subroutine.

The program model will also allow us to study more carefully the difference between instantaneous and non-instantaneous models of acquisition. As Section 3.2.1 will demonstrate, one can show that there is a particular *order* to the way in which certain portions of linguistic knowledge are acquired. In particular, the work of Kean [1974] can be used to show that that the class of possible phonological systems are constrained by a single very general principle of acquisition from positive-only evidence, the Subset Principle. Evidently, hypotheses about new phonological classes are ordered in a lattice of general to specific descriptions, triggered in a particular sequence. This structure is also commmon to several proposed general models of "concept acquisition." For example, this lattice-theoretic structure is also posited in Keil's work on concept acquisition in children, and in models of concept acquisition advanced in the A.I. literature under the heading of "version spaces." Crucially, the developmental model effects a considerable economy of description over an instantaneous model, that is, a model where an entire phonological system is assumed to be "projected" from a body of evidence in a single step.

More generally, there is a close connection between such functionally identifiable blocks of sequenced instructions and succinct representations, hence explanations of data (in this case, linguistic competence). Roughly speaking, a mere list of data in program form looks just like a series of simple statements, one executed after the other, with no subroutines, block structure, loops, or other "higher level" constructs:

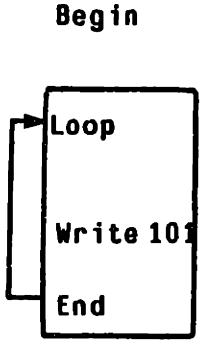
```
Routine to write the string, 110110110

Begin
Write 1
Write 1
Write 0
 etc.

End
```

In contrast, a succinct program, one shorter than a simple list, will have an implicational structure. Such a program will contain clusters of variables akin to condensation points, where values are set and

re-set en bloc.



But as we have suggested, and as will be discussed below, a characteristic of generalizations is that they possess succinct programs. Thus, given the correspondence between succinctness and flow diagram functional units, the program form of such generalizations will contain structural clusters or implicational blocks of code. These implicational clusters, it is claimed, are what psychologists are describing when they speak of observed regularities of development -- in the case of language development, what are often called "stages" of competence. (See in this regard Brown's [1973] discussion of "Stage I" and "Stage II" children.) On this view, temporal regularities in language development are the reflection of underlying implicational program structure, program structure that is in turn a simple fact about the nature of succinct representations. If this view is correct, then it demonstrates that linguistic theory has a crucial role to play in the description of ontogenetic sequences in language development, for it specifies the key points of parametric variation that underlie the developmental program of language growth. Note that if this analysis is correct, then it makes little sense to attempt to account for language ontogenesis without an underlying theory of grammar, just as it is pointless, except as an exercise in naturalistic description, to write down the successive intermediate states of a program's execution sequence without a theory of the underlying program. It suggests also that succinct linguistic analyses that have appeared in the literature may have a more direct interpretation as developmental programs than may have previously been thought.

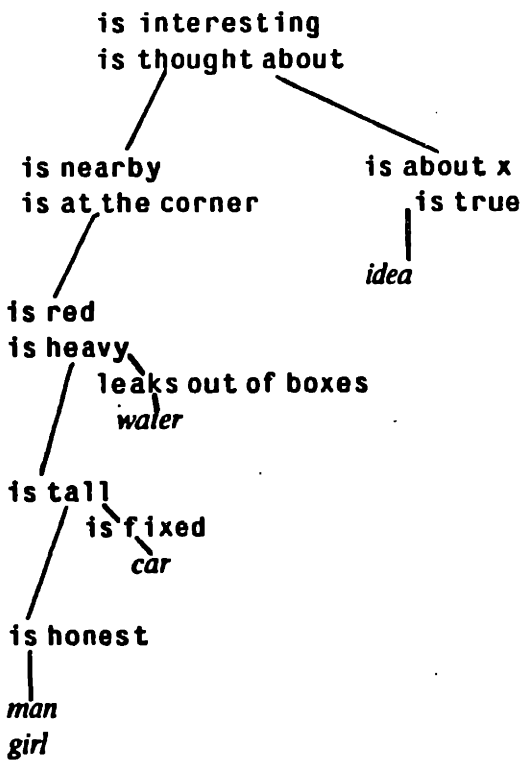
As case studies of this last possibility, this chapter examines two examples, the theory of markedness in phonological rule system developed by Kean [1974] and the X-bar theory of phrase structure. In both cases it will be demonstrated that the information-theoretic structure of the rule system underpins an implicit structure of developmental sequencing, and that as a result both rule systems are best regarded as implicit models of acquisition. More particularly, by interpreting Kean's theory of markedness as a theory of acquisition, we will be able to explain certain of the co-occurrence restrictions on phonological segments that Kean observed. Kean's markedness theory will be shown to fall under a general framework for concept acquisition advanced by researchers in artificial intelligence, the *version space* model developed by Plotkin [1970] and Mitchell [1978], among others.

Third, in addition to providing an explicit developmental model of acquisition based on a theory of grammar, the program complexity approach also allows us to analyze certain formal problems in the acquisition of grammars based on positive-only evidence, what has been called "the logical problem of language acquisition." (Lightfoot, 1981) Two related problems in particular will be considered: (1) The power of ordering statements in acquisition, e.g., the heuristic adopted in the acquisition procedure described in Chapter 2 that ordered the hypothesis of X-bar type rules before the hypothesis of grammar rules, and, within all grammar rules, the hypothesis of local movement before the hypothesis of *Move* α type rules. (2) The difficulty of acquiring disjunctive rule systems, e.g., the fact that verbs such as *want* take both NP and sentential complements. A formal analysis of the ordering principles that have appeared in the recent linguistic literature (see Baker and McCarthy, 1981, for a representative collection of such work) reveals that the proposals that have been advanced are each designed to meet a single necessary and sufficient condition for acquisition from positive-only evidence, a condition that is dubbed the Subset Property. The Subset Property can be used to justify certain other design decisions of the acquisition procedure of Chapter Two, for example, the requirement that specific rules execute before more general rules, or the demand that rule be either all obligatory or all optional; these requirements are entailed by the Subset Property.

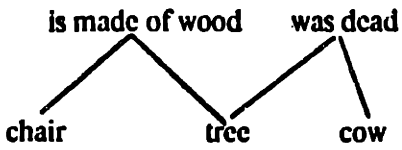
The Subset Property, in turn, is linked to the difficulty of the acquisition of disjunctive rule systems. As will be shown, any rule system that is "disjunctively complete" (in a sense to be made precise below) can generate languages that are not identifiable from positive-only evidence. This result is important for several reasons. First, it means that if acquisition from positive-only evidence is assumed, then the disjunctive brace notation should be eliminated. (This result thus corroborates the work of Winston [1975] who found that disjunctive concept acquisition was nearly impossible in his model of concept learning.)

The elimination of disjunction can be carried out by re-formulating the basic vocabulary of the notational system so as to collapse previously disjunctive categories into single natural classes. For example, as Stowell [1981] suggests, disjunctive complement choices, such as VP-->V {NP or S} might be re-cast as single categorizations, e.g., VP-->V +case recipient. The model presented here shows that Stowell's suggestion has independent support on grounds of acquisition, assuming acquisition from positive-only evidence. The eliminability of disjunction also interacts with the theory of program size complexity and the concept of a natural class itself. Recall that a notational system captures a regularity when it permits one to write short expressions describing long lists of "data." It has often been suggested that disjunctive machinery is inherently at odds with the criterion of simplicity because it in essence allows one to "explain" any set of data extensionally, via a list $S = s_1 \vee s_2 \vee \ldots$. The program size metric shows that disjunction, and hence the brace notation, is indeed non-explanatory as conjectured. A "natural class," then, can be described in one of either two ways: as a description that is more compact than an explicit listing of a set, or as a categorization that is acquirable on the basis of positive-only evidence.

This characterization of "natural class" is also compatible with related research on conceptual development as pursued by Sommers [1971], Keil [1979], and others. Briefly, Sommers has suggested that if one arranges terms such as *dog*, *cat*, *love*, etc. into a graph structure whose terminal leaves are terms and where each node is an item that can sensibly be predicated of the all terms below it in the graph, then one never, or rarely, finds human intuitions of sensibility resulting in "M" shaped graphs -- the so-called "M" constraint. Rather, the graph structures take the form of hierarchical trees. For instance, the tree below (reproduced from Keil [1979, page 15, Figure 1]) is typical of one that comports with human intuitions; an "idea" can be prediated as being true, hence lies under a node labelled "true" in a predication tree, but cannot be at the corner, or red, hence does not lie under nodes labelled with these predicates. In contrast, a car can be nearby, at the corner, red, tall, and fixed, but not true:



Cases of "M" shaped patterns are evidently rare, and are characteristically found in cases of lexical ambiguity, as in the following example provided by Keil [1979 page 13]:



Keil points out that the "M" constraint is motivated on the information-theoretic grounds of compact representation: as observed by Katz [1966], the effect of using a hierarchical network to organize the properties of things such as cars, trees, and ideas is to save space by eliminating the need to list, redundantly, the predicates sensibly applying to both trees and cars, but not both. Keil also suggests that the M-constraint could reflect the demands of acquisition, without being explicit about just how the constraint would help. However, it is plain that the constraint aids acquisition in the following way. "M" shaped patterns correspond to disjunctive, unnatural classes. By avoiding such patterns, an acquisition procedure can ensure that its knowledge can be stated in a disjunction-free fashion -- an important property if acquisition is to be based on positive-only evidence. As noted by Iba [1979] and others, if disjunctive statements are permitted then one cannot always guarantee that acquisition will succeed without appeal to negative examples. In addition, the "M" constraint suggests a specific set of restrictions on the ontogenesis of categorization abilities, as discussed by Keil [1979]. In particular, Keil found through empirical studies of children that the hierarchical tree structures demanded by the "M" constraint developed by non-erasing homomorphic mappings -- that is, children formed new categories by splitting old ones, rather than creating entirely different tree structures.

Intriguingly, the restrictions on phonological and phrase structure systems also seem to obey the M-constraint. As case studies of this topic, Chapter 3 returns to the X-bar theory of phrase structure and Kean's theory of markedness in phonological rule systems. Both systems exhibit the characteristic M-constrained pattern of non-disjunctive categorization, suggesting that these systems, too, are constrained so as to be acquirable on the basis of positive-only information. Moreover, the developmental pattern in both systems is the same, with new categories developing out of old ones. There is no apparent radical restructuring of categorization trees. This confluence of constraint in two quite different rule systems can be taken as strong support for the category-splitting approach to phrase structure acquisition used by the acquisition procedure discussed in Chapter 2.[5] If this analysis of X-bar theory and phonological markedness theory as a model of category ontogenesis is correct, then it suggests a variety of predictions about the externally observable course of acquisition of phrase structure categories and phonological rules. This chapter discusses some of these predictions. For

---

5. In the case of Kean's theory, there are several apparent exceptions to the M-constraint that occur in the case of marked distinctive features such as constricted glottis (as in ʔ). Acquisition of items positively marked with distinctive features such as these would then require negative evidence or explicit correction to acquire -- a legitimate possibility, considering the rarity of the feature. However, the core structure of Kean's system obeys the M-constraint. Alternatively, the presence of M-shaped patterns in Kean's system might be taken as a potential defect that ought to be remedied.

instance, the model subsumes an early and well-known proposal of Jakobson's [1968] regarding the expected order acquisition of phonological categories within a distinctive feature theory. Those categories that appear earliest in the implicational structure of the developmental "program" as defined by Kean's markedness hierarchy are those that must be acquired first. According to Kean's distinctive feature hierarchy, the features *segmental* and *syllabic* would be set first, then the feature *consonantal*. This ordering predicts that *t*, *p*, and *k* would be among the first consonants acquired, and *a*, *i*, the first vowels. Note that this sequencing appears to be *roughly* verified by empirical work, though there has been controversy regarding Jakobson's more restricted, and probably overly-strong proposal. Of course, what developmental sequence is actually observed "on the surface" as it were could depend on the interaction of other factors besides this hierarchy that is presumably a part of universal grammar. In particular, one would expect performance factors to intervene and complicate the actual developmental course. In the case of phonological categories, for example, the delayed appearance of *t* relative to *p* could possibly be due to the intrinsic difficulty of producing a dental consonant -- a matter partially dependent on motor control, hence possibly subject to interference via maturational effects not a part of the specification of UG per se. In this way the actual details of ontogenesis are explained on the basis of a number of interacting, modular principles: (i) a theory of the core portion of phonological categories and rules, as determined by an M-constrained implicational hierarchy, a parameter-setting theory; (ii) a theory of the acquisition of a highly restricted and small number of rules from positive and negative evidence, along the lines of a more traditional theory of induction of finite-state devices from positive and negative evidence; and (iii) a theory of the effects of peformance-oriented maturational constraints.

This same kind of modular account can also be provided in the case of phrase structure rules. Here too, an explanation of observed surface ontogenesis can be broken down into three modular components: a parametric theory of "core" cases, determined by an implicational structure implicit in the X-bar theory itself, with each core X-bar grammar being instantiated by simple positive triggering examples; a small number of marked, "fringe" cases, determined by "brute force" finite-state induction (or, in the case of phrase structure systems, by simple tree automata induction procedures, the hierarchical analogue of the finite state case); and a theory of the recognition complexity for the languages so generated, together with an assumption that stable linguistic systems tend to minimize processing complexity.

As an example of this modular approach, consider the complement structure of English VPs:

VP--> V NP (NP)(PP)* (S) (PP)

Why does the structure of English VPs look this way rather than some other way? Part of the answer, it appears, lies in the existence of certain parametric constraints on grammars: for example, according to the theory of government (Chomsky, [1981]), English NPs that are to receive case from a verb, like

direct objects, must meet a condition of string adjacency. That is, the entire complement structure may appear either to the left or the right of the verb head, but in either instance the NPs receiving case must be adjacent to the verb:[6]

John quickly opened the door.
John opened the door quickly.
*John opened quickly the door.

Examples from Stowell [1981, page 113]

Stowell [1981] goes on to argue that the appearance of a (tensed) S at the end of the VP is also no accident. A tensed S, he claims, cannot be assigned case twice -- once from the INFL element of the S of which it is a part, once from the V outside the S -- and therefore cannot appear adjacent to the V case assigner. Instead, S is extraposed to the end of the VP, as in the following familiar examples:

*John persuaded that the law was unfair the man from England.
John persuaded the man from England that the law was unfair.

Similarly, since Prepositional Phrases are intrinsically case-marked and don't receive case, they may appear non-adjacently to the Verb, and in relatively any order. There is an additional complication in the case of Prepositional Phrases, however, having to do with the way a Verb's obligatory arguments are assembled. For example, in the sentence below the verb *seems* subcategorizes the PP *to us*. Hence this PP must appear in the subcategorization domain of the V. This property is revealed by the fact that an interpolated S *that John is guilty* will seem to block subcategorization, rendering such a sentence ill-formed.

*It seems that John is guilty to us.

In contrast, if the PP is properly placed in the subcategorization domain of the V, then the sentence is fine:

It seems to us that John is guilty.

Thus, PP's can appear after S, but only if they are not strictly part of the argument structure of the V, in some sense. Taken together, these constraints determine most of the observed order of phrases in English Verb complements. Thus, as Stowell argues, there is no need to "learn" this order, which is simply the result of the interaction of the built-in principles of case assignment and subcategorization.

---

6. The requirement of strict adjacency of NPs to the Verb is subject to parametric variation: it is weaker in French, and apparently absent in the so-called "non-configurational" languages such as, perhaps, Japanese. See below for a generalized theory of parsing that can take into account such variants.

It can be argued that the tendency for S to appear at the end of the VP is also supported on grounds of sentence processing. Suppose, as has often been suggested, that people are extremely limited in the number of items that they can hold in short term memory. Then, assuming that the human sentence processor takes a certain form, it has often been noted that grammatical structures that force the storage of a large number of unintegrated constituents in a hypothetical short term memory should be difficult to process.[7] Suppose we say that a Verb is *integrated* when all the arguments it obligatorily subcategorizes have been analyzed. Then the contrast between the following two sentences:

(a) It seems to us that John is guilty.
(b) *It seems that John is guilty to us.

follows from the fact that in sentence (b) at the point where S is being analyzed, two verbs, *seems* and *is* are unintegrated, resulting in an increase in hypothetical memory load. Hence, if the hypothetical sentence processor equates this property with the "integration" of the V's arguments, then it in effect realizes that the first verb *seem* can be dropped from short term memory. Thus the analysis of the S causes less processing difficultly in case (a) than in case (b).

If this analysis is correct, then under the additional assumption that ease of sentence processing contributes to the evolving design of linguistic systems, one would expect systems with terminal S phrases to be stable configurations, and systems with PP--S--PP phrases to be unstable, and much rarer. This prediction seems to be borne out.

Finally, what of the details of the PP^* portion of the complement system? Here there is literally nothing to learn: there is no order to the PPs, nor any "internal" phrase structure, nor any constraint

7. See Chomsky and Miller [1963] and Langendoen [1975] for a discussion and a formal proposal that the human sentence processor can be simulated by a truncated stack push-down automaton capable of handling only an extremely small amount of self-embedding.

on their number.[8]

As Stowell [1981] makes clear, this account of phrase order in natural language represents quite a distinct break from the context-free re-write rule analysis that has been predominant in the literature. Most importantly, it casts an entirely different and interesting light on the problems of acquisition and language processing. In a nutshell, the claim of this revised account amounts to saying that context-free re-write rules perfectly well *encode* the information implicit in the canonical (base) structure of a language, but only as it were epiphenomenally; the context-free re-write notation is the wrong notation, in the sense that it is an artificial theoretical language that only describes the surface facts of phrasal order.

What replaces the constraints that are normally stated using re-write rules? In part this work is shouldered by constraints on Case assignment -- as, for example, in the case of the structure of English Verb complements, described above. In addition, Chomsky [1981] suggests that a powerful source of restrictions on possible surface phrasal form comes from the principle that the subcategorization frames of lexical entries -- e.g. the fact that *persuade* takes an NP and S complement -- must be obeyed

---

8. Note that at first glance there would seem to be a problem in deducing that the allowable complement structure of VP is of the form NP--(NP)--PP^*--S while in the midst of parsing, because it might well be that a PP is part of an NP complement, as in, *kiss the man with the frown*. But this is not a problem if one abides by the restrictions of X-bar theory. According to X-bar theory the complement structure of *any* category, V, N, P, or A, determines the complement structure of all the other categories – in essence because in the unmarked case there aren't any separate phrase structure rules for Noun Phrases or Verb Phrases. But then whether a PP is chosen to be part of an NP or the VP is irrelevant; no matter which choice is made, the acquisition procedure has positive evidence for the existence of PP's in the complements of VPs and NPs. From another point of view, though, even this account is incorrect. It is simplest just to say that PP's can appear anywhere, optionally, and that nothing about them needs to be acquired from external evidence, positive or otherwise. Then the appearance of PP's in between NP objects and S complements is just a restriction that surfaces as a result of other constraints, namely that of case assignment. (The highly marked status of the S–PP order would still be expected on the basis of its processing difficulty.)

Put another way, it seems as though part of the phrase structure component of grammars for natural languages are constrained to be *non-counting*, in the sense of Crespi-Reghizzi *et. al.* [1978]. That is, grammars for natural languages do not indicate in phrase structure, for example, that three PP's may appear in a VP but not four or five. Rather, the grammars for natural languages are *free*. A language wrt grammar G is *free* just in case vwu in L(G) implies that $v^{n+m}\, w\, u^{n+m}$ is also in L(G). A grammar (or a particular phrase structure rule) is free just in case it generates a language that is free. For example, the VP rule VP-->V NP $(PP)^*$ S is free with v= null, w= V NP, and u= PP, now regarding the non-terminal categories V, NP, and PP as "terminal" elements in a meta-grammar for the phrase structure grammar actually instantiated.

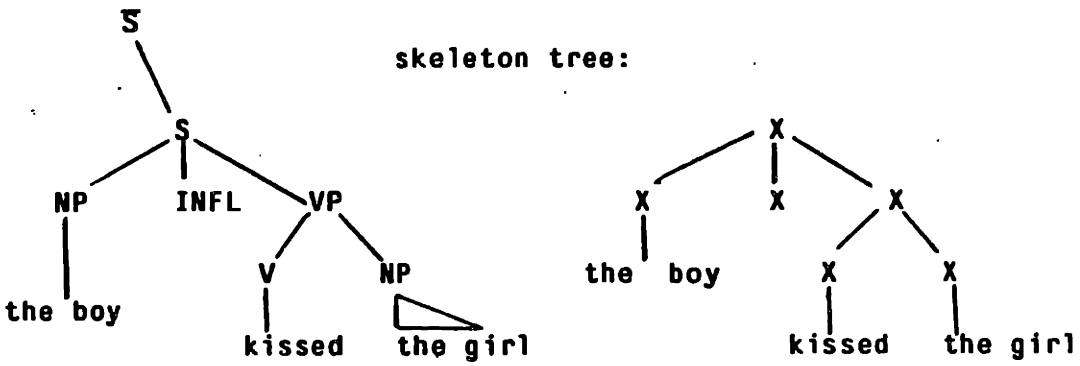
This restriction appears to be too weak, however, since a VP cannot have an arbitrary number of V's or S complements. Apparently, the restrictions of case assignment are such that the subcategorization frames for Verbs *can* mention that there are a particular number of complement phrases of a given type. For example, the subcategorization entry for *persuade* demands one NP and one S complement, but not two NP's. However, this is knowledge is part of the lexical entry associated with *persuade*, not phrase structure per se.

Since PP's are intrinsically case marked they should not be bound by these counting restrictions. Thus PP's should be free -- as is the case. This is important from the standpoint of acquisition, since the assumption of non-counting allows one to infer from the observation of just a finite sample of data that the right expansion rule for PP's is PP^* rather than, say, P^{10}.

at the level of surface structure as well as at the level of lexical entries themselves. More generally, this principle can be extended to the claim that the structure of lexical entries is *projected* through every linguistic level of representation, the so-called Projection Principle:

> Representations at each linguistic level (LF [logical form], and D- and S-structure, are projected from the lexicon, in that they observe the subcategorization properties of lexical items.
> [Chomsky 1981 page 29]

The Projection Principle and constraints on Case assignment work together to determine the types and order of phrasal constituents. Once the constraints they provide have been factored out, it is not clear that anything like traditional phrase structure rules are required to account for the observed patterning of constituents. From the standpoint of acquisition, to construct a context-free re-write rule we need to know two things: (i) the non-terminal labels of the rule, e.g., the labels NP, V, S, and the like; and (ii) what the phrase boundaries are.[9] The basic point of X-bar theory is to essentially eliminate category labelling, aside from the "projections" of lexical items. That leaves the problem of phrasal boundaries, or the "skeleton" structure of phrases:



A theory for the induction of phrase structure labels given only skeleton tree structure has been advanced (Joshi and Levy [1977a]); it is a straightforward generalization of the induction of a minimal

9. Compare Hale's [1979] statement that phrase structure rules provide category and hierarchy information. Plainly, this *exhausts* the information that a context-free re-write rule can specify.

finite-state machine given only the "black-box" input-output response function of that machine.[10]

There is, however, one major problem with this approach to the induction of phrase structure: it is data intensive. In fact, as is well known, if the "correct" minimal phrase structure grammar for the language to be identified has n distinct non-terminals, then the brute-force induction procedure must examine all (positive example) trees of depth $\leq 2n+1$ -- in general, an overwhelmingly large number of data samples. [11]

Suppose then that we abandon completely the view that surface phrasal order is acquired via a general kind of induction procedure, or even that anything like phrase structure rules are acquired in the course of language learning. In this revised account, what is central to what we observe on the surface as phrasal co-occurrences are the core notions of government, verb subcategorization, and case assignment; there are no "phrases" as such, except as they are defined by these principles. (These terms will be defined more carefully below.)

From the standpoint of sentence processing this proposal suggests a quite different point of view than that usually studied in the computational literature. Most parsing procedures for "natural" grammars contain a substantial phrase structure component, typically involving many hundreds or even thousands of re-write rules.[12] Quite naturally then, in these procedures non-terminal categories and rules play a prominent role, acting to define the states of a parsing machine that analyzes sentences in a specified way. (In the Gazdar theory this approach is carried to its extreme, with non-terminal categories being used to encode almost all aspects of co-occurrence restrictions.)

This approach leaves us with two major questions. First, how is parsing to be done if we cannot avail ourselves of phrase structure rules? Second, parsing is an interesting computational problem only insofar as phrasal structure is rich and highly developed. In languages such as Japanese, phrase structure appears relatively "flat" and uninteresting, a collection of NP's ending with a V that looks

---

10. The proof uses the hierarchical generalization of finite state automata, tree automata, to construct Myhill-Nerode equivalence classes of internal states of a machine -- hence grammar -- having the same response function as the observed "black box" language. Roughly, states are labelled via constructing strings that "behave alike" in that states that are equivalent map such strings into already known to be equivalent states. The base step of this recursive procedure are the accepting and rejecting states of the machine, states that are known to be non-equivalent. Each non-terminal of the final grammar represents a different equivalence class.

11. That is, $2N+1$ positive examples *suffice* for the induction. It might not be true that $2N+1$ examples are necessary, however. In fact, in restricted situations, such as the X system, one need identify only three "nonterminals" -- namely, the Head, Specifier, and Complement of a rule. Even so, if the requirement of Degree-2 induction is to met, then we must ensure that all differences in phrase structure are evidenced on sentences with at most two embeddings. Note that we can avoid this problem if we say there are in effect no phrase structure rules at all, or just one "category" of type X. See Section 3 for additional discussion.

12. This is true even of proposals as diverse as the Bresnan-Kaplan lexical-functional grammar [1980] and the Gazdar "slashed-category" grammar [1981].

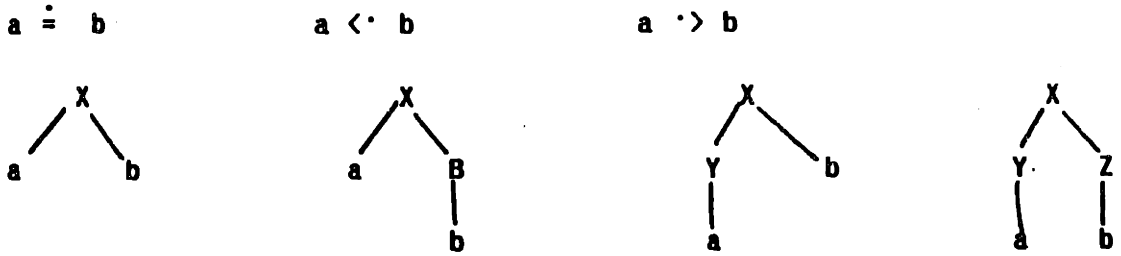
very much like the "flat" arrangment of PP's in English complement structure. Still, Japanese possesses a rich set of co-occurrence restrictions -- but one not based on phrasal structure. How then can one develop a parsing procedure that works with languages that are not keyed to the existence of configurational structure, but are, rather, neutral with respect to how grammatical relationships are encoded? Such a parsing procedure would act as if it were phrasally-based in a highly configurational language, such as English, but would also act as if it were not hierarchically oriented in a non-configurational language.

The dual demands to abandon explicit phrase structure rules and accommodate non-configurational languages into a general parsing model, are, it might be suggested, two aspects of one and the same problem -- the non-existence of an explicit representation of phrase structure . What we seek as a replacement is a model that parses without using context-free re-write rules at all. Such a model would reconstruct the notions of association and phrasal boundaries inherent in re-write rules at a higher level of abstraction, incorporating both configurational languages (such as English) and non-configurational languages (such as Japanese). Non-terminal categories would play little or no role, and phrase boundaries would simply be the "reflexes" of the more fundamental principles of government and subcategorization. Is such a model of parsing at all plausible?

Perhaps surprisingly, the answer may be yes. One of the oldest and most primitive methods of parsing, in fact, was based precisely on a model that in essence *ignored* non-terminals (hence context-free re-write rules), acting solely on the basis of properties of terminal input symbols and a set of association or *precedence* relations between terminal symbols. (Floyd [1963]; Wirth and Weber [1966]) Intuitively, in a precedence model of parsing boundaries of phrases are located by looking up in a table whether two terminal items *a* and *b* are to be lumped together in the same phrase, or whether the "boundary" between the two items indicates the start of a new phrase, with either *a* dominating *b* or *b* dominating *a*. (Note that this exhausts the possibilities.) If *a* and *b* are in the same phrase, they are said to be of *equal precedence* (denoted a $\doteq$ b); the terminology comes from the familiar domain of parsing arithmetic expressions such as 2+3*5+6. Roughly, if *a* Constituent commands (c-commands) *b*, then *a yields precedence to b* (denoted a $\lessdot$ b) and *a* indicates the left-most end of a phrase, where α Constituent commands β iff the first branching node that dominates α dominates β. If *b* c-commands *a*, or is derived from a constituent that c-command *a* then *a takes precedence over b* (a $\gtrdot$ b) and *a* indicates the rightmost boundary of a phrase:[13]

---

13. Formally, the Wirth-Weber precedence relations for a CFG are defined as follows:
X yields precedence to Y if there exists A-->αXBβ such that B-->$_+$ Yψ. X is equal in precedence to Y if there exists a rule A-->αXYβ. X takes precedence over a terminal *a* if A-->αBYβ, Y-->$_*$aγ, B-->γ $_+$X



Note that while the precedence relations might be entirely determined by the phrase structure, hence the re-write rules of a grammar, there is nothing inherent in the approach that demands that they be so specified. For example, in arithmetic examples such as "2+3*5+6" it is quite common to specify the precedence of the operations "+" and "*" by reference to an arbitrary "semantic" rule that multiplication takes precedence over addition. It is this flexibility that is attractive from the point of view of natural grammars. If the precedence approach can exploit the relationships implicit in the principles of government,case assignment, and the like, then one might be able to obtain a general-purpose parsing method that would be instantiated as a phrasally-based procedure in highly configurational languages, and a (morphologically? based) procedure in non-configurational languages.

Turning then to the domain of natural grammars, suppose that phrase structure rules are indeed ephiphenomenal. Let us replace the bracketing imposed by phrase structure rules by the parenthesization implicit in the precedence relations demanded by, for example, the core notion of a Verb governing an NP argument:

John $[_{VP}$ [V- kiss] $[_{NP}$Mary]]

In English, it is conventional to say that in sentences such as this there is a Verb Phrase, the maximal projection of V in X-bar terms, headed by the Verb *kiss*; the NP *Mary* is in the Complement of the VP, assigned Objective Case by the Verb. Suppose though, that we turn this way of looking at things on its head. Instead of saying that a rule such as VP--> V NP exists, let us say that a (Verb) phrase is determined by the precedence inherent in the government structure V--NP, now interpreted as a kind of operator-operand structure. Then just as the precedence of the multiplication operator determines that a+b*c is to be "phrased" as a+(b*c), the government defines a phrase called a "Verb Phrase." What if the governed NP is not adjacent to the Verb? This is apparently not permitted in English, but it is in languages possessing VSO structure, such as Arabic. Noting this, Aoun [1979] has proposed that V--O forms a kind of discontinuous VP in such languages, implicitly defined by grammatical

principles. Another example is found in discontinuous Verb-Particle constructions in English (e.g. *John picked the ball up*). This approach amounts to the proposal that an abstract Verb Phrase exists -- as defined by government -- but that there is no *rule* of the form VP-->V... From this point of view, context-free re-write rules are simply a notational system that can perfectly well *describe* surface constituent structure order, but they are not representational objects that are literally engaged in mental computation. The notion of *phrase* is derivative, being parasitic on the primitives of grammatical precedence (government) and adjacency.

In Section 4 of this chapter this proposal will be examined in more detail. As will be shown there, the X-bar system appears to specify a narrowly restricted subset of the class of possible context-free grammars. X-bar grammars can be interpreted as operator precedence grammars, where the core notion of "government" defines precedence, and the relationship of lexical Heads (e.g., Verbs, Nouns, Prepositions, and Adjectives) to their complements determines an operator structure. This result is intriguing from the standpoint of acquisition and sentence processing:

> (1) It accomodates both configurational and non-configurational languages in a single, generalized parsing model based on the Marcus design (a restricted, two-stack parser, with input buffer acting as one stack, and active node stack as the other). For example, such a design seems to be able to handle non-configurational languages such as Japanese.
>
> (2) Precedence parsers can be designed that build a "skeleton" tree, indicating just bracketing structure but not category identity (non-terminal labelling). That is, such a parser operates without having to identify the identity of a phrase as a VP, AP, etc. -- in other words, as if there were just a single sort of phrase structure rule, XP--> ...X...
>
> (3) By using a precedence matrix instead of an explicit context-free grammar to define phrases one can naturally incorporate the resolution of structural ambiguity via extra-syntactic decisions. (Just as the precedence of multiplication over addition is determined "arbitrarily".) This approach also allows one to adopt Weinberg and Hornstein's proposal [1980?] regarding the "reconstruction" of Verb--Prepositional forms as complex predicates (as in, *John* $[_V$*slept*$]$ $[_P$*in*$]$ *the bed*--> *John* $[_V$ *slept -- in*$]$ *[the bed]*). Reconstruction receives a natural interpretation in the precedence model as redefinition of the V "operator" to a V--Prep form.
>
> (4) With regard to acquisition, it is easy to show that there are but a *finite* number of precedence grammars that impose the same structural descriptions on sentences of language; this is because the possible matrices of precedence relations is finite. Hence, if X-bar systems can be expressed in terms of precedence matrices, then there are a finite number of such systems. This alters considerably the purely mathematical difficulties of

acquisition. (As observed by Crespi-Reghizzi [1970], since the class of free grammars cannot generate all finite languages -- languages with a *particular* number of constituents, such as A^7, are not generated by free grammars -- the theorems of Gold [1967] demonstrating the inability to identify context-free languages on the basis of positive-only evidence do not apply.)

(5) The precedence model clarifies the possible role of other sources of information in the parsing process, in particular the contribution of metrical structure constraints as revealed by stress contours. If one interprets "parsing" as the recovery of a representation of annotated surface from surface perceptual information, then, as we shall see, some of the precedence relationships defining annotated surface structure can be deduced from stress contour information alone. (In general, however, metrical representations are not isomorphic to annotated surface structure representations, as observed by Rochemont and Culicover [1980].) This use of metrical structure may play a role in acquisition as well, defining precedence boundaries in the presence of incomplete information.

(6) Certain properties of early child language, in particular so-called "pivot" productions where a certain fixed terminal operator (such as a specific verb) appears in conjunction with a variable argument (such as an NP) are easily seen to be subsumed by a precedence model.

To summarize, the "parametric" X-bar account of acquisition is to be contrasted with the view that the phrase structure rules for a particular language are built up by an incremental process of inference and generalization from observed examples, perhaps given certain additional evidence such as the correct placement of constituent boundaries. The "observe and generalize" approaches have most often been advanced as part of the study of the inductive inference of languages as described by Solomonoff [1964] or Feldman [1969] (and originally suggested in Chomsky and Miller [1957]). The core idea of inductive approaches is to hypothesize new phrase structure rules by merging old rules with new evidence; for example, an old rule X--> Y Z and evidence indicating the existence of the expansion rule X--> Z might be merged to form a modified rule, X--> (Y) Z, indicating the optionality of the component Y. Formally, as was mentioned above, this process is one of determining the equivalence classes (=non-terminal labels) of a machine on the basis of observing its input-output response pattern. Using the terminology of pattern matching, the new, merged rule is the "most specific generalizer" (MSG) of the other two rules. (This method is also common to a popular approach developed by Artificial Intelligence researchers for the induction of general rules from specific examples known as "version spaces.") More recently, a variant of the MSG approach has been suggested as a procedure for the acquisition of the phrase structure rules of the theory of lexical-functional grammar (Pinker [1980]), who advances a set of generalization heuristics for phrase structure rules much like those suggested in the references cited above.

What MSG models share is the assumption that some of the burden of acquisition may be shifted from restrictions on the class of possible grammars to the capabilities of some kind of generalization mechanism -- the rules that decide how to merge old phrase structure rules into new rule. It has further been suggested that the possibility of "enriched" input to an acquisition procedure, such as placement of constituent boundaries, also lessens the need for a rich set of initial constraints on possible grammars. In effect some of the constraint required for acquisition is to be displaced from UG-type restrictions to restrictions on the order and kind of input.[14]

However, from the perspective of the model advanced here, both of these alternatives are misguided. There is no set of heuristics for unifying the description of phrase structure rules, because there are no phrase structure rules; there are no decisions to make about optionality, disjunction, or repetition of constituents because constituents may be freely repeated and are optional.

In fact, as suggested earlier, the MSG approach can be shown to be the hierarchical analogue of the problem of inducing the state diagram of a finite state automaton on the basis of observing the external behavior of that automaton.[15] It is thus subject to all the formal difficulties of finite-state inference procedures. Specifically, it is shown below that models that assume that some kind of general inductive procedure plus the availability of constituent boundary information is sufficient to guarantee learnability fail to take into account the large amount of external evidence that must be supplied. For one thing, trees of large depth are required -- depth about 2N, for N non-terminal categories. This seems decidely unrealistic. In fact the amount of external evidence required grows exponentially in the size of the grammar, where the size of the grammar is measured by the number of distinct non-terminals it contains. This result suggests that the notion that phrase structure rules are *generally* induced on the basis of external evidence is faulty. Two remedies have been suggested in this thesis: the first is to abandon induction as a workable method of rule identification, except in tightly restricted systems, such as the Inflectional rule system of English. This was the approach used in Chapters 2 and 3.

To summarize this introduction, the overall aim of this chapter is to justify in a formal setting the design of the acquisition procedure described in Chapter 2. A formal theory of markedness will be presented assuming a framework of acquisition from positive-only evidence, and this demand, in turn, leads to the justification of several of the ordering principles posited for the acquisition procedure.

Section 2 of this chapter outlines the theory of program size complexity, as developed by Kolmogorov, Chaitin, and others, and shows how it subsumes the markedness theory proposed by Kean. This

---

14. See Wanner and Gleitman [1982 forthcoming].
15. See Joshi and Levy [1977a] and below.

section also reviews the M-constraint and its implications for acquisition.

Section 3 examines the conditions necessary for acquisition from positive-only evidence. It shows that a single characterizing condition, the Subset Principle, subsumes all acquisition ordering constraints that have been proposed in the linguistic literature -- see Baker, Lasnik. This result is not surprising, since the Subset Principle is a necessary condition for acquisition from positive-only evidence.

Section 4 reviews the theory of X-bar acquisition from a formal standpoint. It first discusses the possibility of acquisition via brute force induction of categories, concluding that the obvious tree automata approach is too data intensive. This approach is justifiable only in restricted, finite subsets of phrase structure, e.g., the AUX or INFL system.

3.2 Program size complexity and acqusition

The theory of program size complexity as developed by Kolmogorov, Chaitin, and others, takes the *complexity* of a set S with respect to a programming language Λ to be the size of the smallest program written in language Λ that outputs S. The relationship of this approach to the framework advanced in MPMH should be evident. The set S corresponds to a set of strings of some linguistic level -- a string of phones, phonemes, syntactic categories, etc. The programming language Λ (together with any associated interpretive procedure for "executing" statements written in Λ, i.e., for deriving strings of level L) corresponds to the notational system itself. (Recall the formal connection between re-write rules, derivation sequences, and computation sequences of Turing machines.) Thus a given rule system denotes a program for a set of sentences of a linguistic level. Clearly, there is nothing to prevent us from applying these notions to any set whatsoever. In particular, we may apply it to the case where S is a set of rule systems, i.e., a family of grammars, and the programming language A is a notational system for describing grammars -- a theory of grammars. A program for a particular grammar G_i then takes the following form: given a collection of grammars G, a program for G_i "reads" some input (perhaps null) and "writes" as output a representation for G_i. The complexity of G_i is then defined as the minimum size such program, among all programs for G_i.[16]

More formally, let A_i be a partial recursive function (prf). A_i maps strings to strings, (X^* to X^*), which we may take to be the coded representations of the sets we are actually interested in. Let p denote a program written in language A_i, and |p| the size of p. $A_i(p)$ denotes the output produced by running p. Then the Kolmogrov complexity of a string s wrt A_i is defined as:

---

16. Note that as demonstrated by Blum [1967], the general problem of determining the minimal size program for a given set, given the full power of an acceptable programming language, is not recursive. Presumably, however, we are not dealing with the general situation that Blum considers.

$$K_{A_i}(s) = \{ \text{ minimum } |p| \text{ s.t } A_i(p) = s$$
$$\text{undefined, otherwise } \}$$

Consider by way of example the string, *101101101101*, and let A_i be the prf that simulates a common programming language, say Fortran. There are at least two Fortran programs for writing out this string. One simply lists all the bits of the string:

```
WRITE(6) '1
WRITE(6) '0
WRITE(6) '1
 etc.
STOP
END
```

This program will be about as large as the length of the string itself -- assuming that each instruction occupies one memory location. (There will be a small additional amount of storage occupied by format statements, a STOP and END instruction, and so forth.) Plainly, there is another, shorter program for writing out this same string:

```
DO 10 I=1, 4
10 WRITE (6) '101
STOP
END
```

This program occupies only four memory locations -- plus a register to hold the loop variable I. In fact, as the number of 101's gets larger and larger, the second program will increase in size only as log(n/3), where n = the length of the string. Assuming that this second program is as short as any other for this string, then the complexity of the string is four. We see then that the string *101101101101* has a program description shorter than itself. Intuitively, this is because the string contains a *regularity* that can be captured by program code that does not merely list the elements of the string. Still other strings of 0's and 1's cannot be so compressed, because they have no such regularities. Intuitively, these correspond to precisely those that are thought of as being patternless, or

random. (Consider writing a short program for the string *10100011O.*)[17] This intuition inspires the following standard definition:

A string *s* is <u>random</u> if $K_{\Lambda_i} > |s|-c$, c some constant.

The next issue to address is the question of the choice of notational system. As described in the introduction to this chapter, if the prf chosen is a universal prf (a Turing machine program that can simulate all other Turing machine programs, i.e., all other prf's), then the complexity of a string wrt such a universal notation is at worst a constant different from the complexity of that string measured wrt any other prf (notational system) Λ_i (not necessarily the same constant):

$$K_U(s) < K_{A_i} + c$$

It is immediate that the complexity of a string measured wrt any two universal notations will differ from each other by at most a constant, namely, the maximum of the constants used to simulate one program via the other. Thus we may replace the particular Λ_i used in the definition of a random string above by an arbitrary universal notational system, without affecting things. Put another way, a random string has no expressible regularity in *any* universal notational system. The only way in which a random string can be written out is to encode the string in the program itself, and then have the notational system to act like the identity mapping, $\Lambda(p) = \Lambda(s) = I(s) = s$.

Turning now to the domain of linguistic theory, recall that the goal of modern linguistic theory has been to characterize the final states of linguistic knowledge such that these states are "projectible" from what human initial states of knowledge must be, under the conditions of data input, evidence, and such that are encountered. In the program model, the output string produced corresponds to this final state of knowledge, a grammar; the initial state of knowledge is simply the information available at the start of the computation -- namely, initial tape contents plus the program itself; and the projection function is implicit in the prf denoting the machine on which the program "runs." There are also two distinct ways in which one might imagine the program receiving the input data it requires: (i) the data may be written on the program's input tape to begin with; or (ii) the data is presented sequentially over time. Below we consider whether these different methods of data input actually make a difference in a characterization of possible developmental programs. In any case, by the usual

---

17. There is a well-known approach for information storage and display that is but a special case of program size compression: so-called *run length encoding*. In this technique, a long string of constant values (a run of 0's or 1's, say) is encoded via two numbers: the number of repetitions and the number that is to be repeated. Thus the string "101 1" would denote a string of 5 1's. If the number of changes from 1 to 0 or 0 to 1 is small -- that is, if in general the data consists of long runs -- then the compression achieved by this method is considerable. A string of *n* 1's and 0's, where the runs are of O(n), will take O(log n) storage. At the other extreme, if the string consists of an alternating pattern of 0's and 1's, then the straightforward encoding expands the string, since every 1 is encoded as 11, and 0 as 10. Note that in this case a more global compression succeeds: the number of 10's suffices to store the entire string.

poverty of the stimulus type arguments, the amount of such external information is assumed to be small -- hence the goal to keep this information to a minimum, compatible with the observed range of possible human grammars.

From this standpoint, a strict program size measure is not quite the right one. For if acquisition complexity really measures the difficulty of selecting a grammar given input evidence about the language generated by that grammar, then a grammar could be very large and yet acquisition could be extremely easy, even trivial. This might be the case if, for example, there was but a single human grammar. Then literally no external information would be required to identify the grammar; it could be entirely "built in," with no obvious size limitation. Such a system would have in effect zero acquisition complexity . Similarly, suppose there were two possible grammars. Then, following the usual definition of information content, the complexity of this system of grammars is $-\log_2$ (# of alternatives) = -1. Roughly, one decision suffices to choose between the two alternatives. (Of course, in a cognitive context this decision must be made on the basis of evidence available to the child.) Let us call this complexity measure *developmental complexity* (DC), and the specific value of DC associated with a grammar measured relative to some system of possible grammars its *markedness*.[18] According to a developmental measure, we are to penalize a system that requires a large number of decisions to select the proper grammar, given some initial distribution of possible grammars. This measure is thus a proxy for the cognitive demand of poverty of the stimulus. Put another way, the developmental measure attempts to minimize program size, but where the "program" is now the developmental program that starts from an initial set of possible grammars and selects a correct final state grammar corresponding to the linguistic competence of the adult community. Why then isn't the right grammar always completely encoded in the initial state of the system to begin with, thereby reducing decision complexity to zero? For one thing, as has frequently been observed, the developmental system must retain some flexibility, because the child cannot know in advance what language community it will be a member of. The "open parameters" of the initial state must be such that they cover the range of variation of possible human languages.

Second, and more speculatively, the existence of parametric variation that is fixed by external developmental decisions may reflect the advantage of placing some of the information-theoretic burden in developmental sequencing rather than completely in the initial state of the system itself. To see why, first observe that there is a close connection between initial program size and developmental complexity. For example, consider the following three hypothetical example grammatical systems that aim to specify the structure of a Verb, Noun, Adjective/Adverb, and Prepositional Phrases, where a

---

18. To my knowledge, this distinction between *descriptive* and *developmental* complexity first appears in Solomonoff [1964]. Solomonoff also proposed a joint trade-off function between program size (descriptive) complexity and developmental complexity, as discussed below.

Phrase is assumed to consist of a Head (Verb, Noun, Adjective/Adverb, or Preposition), preceded or followed by a Noun Phrase, a Prepositional Phrase, and an optional S, in any order. (We are therefore assuming that X-bar cross-category generalizations do not hold.) There are 12 possible arrangements of constituents forming Verb Phrases; in all, there are 12^4 possible rule systems. Suppose that System A explicitly stores all possible systems (as part of UG), selecting the right system based on some kind of triggering evidence that is entirely unrelated to the linguistic system itself. Thus for the Verb Phrase alone it will have to store 12 possible expansion orders. System B stores just the unordered set VP-->V{NP, PP, $\bar{S}$} or VP-->{PP, NP, $\bar{S}$}V, and fixes the right order via external evidence. Plainly, in the worst case this may take six data samples (e.g., V PP, V NP, V $\bar{S}$, V NP PP, V PP $\bar{S}$, V NP $\bar{S}$). We may represent B's developmental program schematically as follows:

Fix V-Complement order

V-Complement	Complement-V
V-PP	PP-V
V-NP	NP-V
V-$\bar{S}$	$\bar{S}$-V

Fix PP-NP-$\bar{S}$ order

Note how program compression is achieved in this case. Suppose that Head-Complement order is fixed for a Verb Phrase by a single example sentence. We can now set the remaining order by just three additional examples, no matter what the outcome of the Head-Complement decision. From a slightly different point of view, this compression is evidenced by the fact that there is a regularity about phrase structure that can be couched in terms of a Head-Complement metavariable. In either case, instead of 12 separate decision points, one for each possible VP system, we have obtained a more succinct program representation by breaking the selection down into two modular steps, one that consists of two possible choices, the other, of six possible outcomes. Thus the storage for the modular, program representation is of size approximately $(6+2)=8$, smaller than a list of the 12 possibilities. In this way a multiplicative number of developmental possibilities can be stored in an additive amount of space, simply by cascading developmental decisions over time. In short, by representing the selection of a grammar as occurring over time via a sequence of decision points, we have collapsed the description of the grammar. In effect we have exchanged the size of the system that must be stored as part of UG for an increased number of developmental decisions. Note that the total amount of

information required to fix a grammar remains constant; all that changes is *where* that information is allocated.[19] This trade-off might, then, be used to reduce the size of UG, assuming now that the space available for specifying UG is limited.

The "compression" inherent in the definition of an abstract super-category such as a Complement or Head does not exhaust the possibilities for program size improvements in phrase structure acquisition. The system that A and B were attempting to describe was assumed not to obey the X-bar regularities. Thus the structure of each type of phrase would have to be determined separately, four decisions for each of four phrase types for a worst-case total of 16 decisions (or 6 examples for each for 4 phrase types, for a total of 24 data points). But suppose the X-bar regularities do hold. Then the structure of Verb Phrase complements may be assumed to predict that of Prepositional, Noun, and Adjective/Adverb Complements -- the X-bar cross-categorial systematicity. This regularity is reflected in the reduced developmental program size required to fix an X-bar type grammar. Only four decisions (6 samples) are required to set the rule system for all phrase structure types. Equivalently, one may say that there are no separate phrase structure rules for VP, NP, PP, and AP; rather, there is just one non-terminal class of type X. We see then that in this case informational redundancy, the fact that VP Complement structure predicts that of Complements of all categories, is reflected as developmental simplicity.

There is one other modification to a grammar that may impact both descriptive and developmental complexity, and that is the re-definition of phrasal categories themselves. For example, as Stowell [1981] observes, NP and S often appear in alternation in the complement of a VP:

I believe John.
I believe that John is a fool.

Two developmental decisions must be made to arrive at this grammar, one to determine that NP may appear in the VP complement, and another to determine that S may appear. We can reduce the size of the grammar and at the same time the number of developmental decisions that must be made if we collapse these two categories, NP and S, into one. In this case, Stowell suggests that NP and S are members of the same natural class, sharing the property of being Case Recipients -- NP is Case-marked by the Verb, and S by the inflectional element of a sentence. Suppose that this suggestion is correct. Then we have shortened the description of the Verb Complement from,

VP--> V { NP or $\overline{S}$}

---

19. See Chapter 4 for a formal model of the succinctness possible with "modular" descriptions and developmental systems of this kind.

to

VP--> V Case Recipient

Observe that this was the approach adopted by the model described in Chapter 3, where the feature ±*Argument* was used instead of the feature *Case Recipient.*

Let us characterize this situation more formally. Fix some set of possible grammars. Now suppose that P is the shortest program for grammar G_i of this set. We may imagine P's initial tape having written on it a sequence of input data, corresponding to a valid sequence of "external" data samples, sentences of the language of G_i, if we are assuming positive only evidence. The control table of P holds the program that, in conjunction with the initial tape contents, is sufficient to fix G_i: P moves through a sequence of decision states, $D_1,...,D_n$, outputting a representation of grammar G_i. (perhaps the grammar itself, perhaps an index for the grammar).[20] Then P is just a program for computing G_i, and thus we may say that the descriptive complexity of G_i is just |P|. A decision theoretic model can in this way be converted into a description of a set. Now suppose there is a program P^* that also fixes G_i, but using a smaller number of decisions prompted by external evidence. By assumption P was already as small as possible. So if we reduce the external evidence required to fix G_i (as measured by the contents of the input tape), then the control table of P^* must be larger than that of P. More simply, the information required to specify G_i remains constant, and all we have done is to shift some of the burden from external data to initially encoded program.

3.2.1 Kean's theory of phonological markedness: an application

To summarize the mostly abstract discussion so far, markedness has been defined as the amount of information that it takes to fix a particular grammar (or rule in a particular grammar). It was further claimed that the information load required to fix a grammar could be reduced by the right combination of initial constraints and developmental sequencing. But are these abstract possibilities actually realized in practice in an actual linguistic system? Is the model of program size a useful one in the study of acquisition complexity? As a concrete case study this section examines in detail Kean's [1974] theory of markedness for phonological segmental systems and phonological rules.

According to the distinctive feature theory of phonology originally developed by Prague school structuralists such as Jakobson and pursued by Chomsky and Halle in *Sound Pattern of English* [1968], all natural sound segments such as *a* or *p* can be described via a small number of binary-valued

20. Note once again that there may be no recursive procedure to find P, in general.

distinctive features. By and large these features have an articulatory or an acoustic grounding, with names suggestive of their place or manner of articulation, such as *high*, *back*, *anterior*, and the like, 20 or so features in all. Given binary values for distinctive features (+ or -), there are 2^{24} possible segments (about 16.8 million). However, most of these segments are not attested in human phonological systems. Furthermore, most *collections* of segments in a particular natural language do not make use of an extensive number of these possible contrasts.

As Kean shows, in part the reason for this is that distinctive features are not fixed in isolation. Rather, certain distinctive features can be fixed *only after* certain other features are fixed. For instance, according to Kean's theory the distinctive feature *consonantal* must be fixed before the feature *back* or *continuant*.

Kean developed this theory as way to explain some of the observed restrictions on possible segmental systems and possible phonological rule systems. But there is another way to interpret such a theory, and that is as a developmental program for how a segmental system is acquired. The hierarchically-organized implicational structure involved in setting a particular distinctive feature value is in fact a developmental "program" of just the sort described abstractly in Section 1. That is, we can identify the *markedness* of a particular rule, segment, or phonological system as the amount of information required to fix that system developmentally. This structure in turn permits a considerable compression of the information required to fix a particular segmental system, just as the program metaphor suggested. For example, as we shall see, instead of demanding a selection of the distinctive feature value settings of *a* out the entire space of possible settings (about 2^{20} of them), the developmental program can fix *a* by setting just eight parameters.

By construing the theory in this new way, one can in fact exhibit an acquisition system in which large numbers of developmental pathways are eliminated because of the *order* in which a small number of parameters are set. Moreover, the developmental model actually *explains* the existence of Kean's constraints, constraints that, while presumably universal, are otherwise not accounted for. Put another way, taken as a model of acquisition the Kean theory provides an explicit example of how developmental sequencing can actually be an aid for acquisition, above and beyond the constraints provided by models of acquisition that assume an "instantaneous" fixing of a system of knowledge.

To see how this approach works in detail it will first be necessary to outline Kean's theory of markedness for phonological segments. Kean states the basic aim of her theory as follows:

> It is assumed here that there is a relatively small set of distinctive features with binary specifications in terms of which all the members of every segmental system can be characterized at every stage of phonological representation. The postulation of such a set of features makes a substantive claim as to the class of possible elements in phonological systems.

> Of the set of possible segments characterized by the distinctive features, it is evident that some are present in nearly every language, with others only occasionally occurring. For example, the segments t and a are nearly ubiquitous in segmental systems; they are found at all stages of phonological representation in an overwhelming majority of languages, but the segments kp and u only occasionally enjoy a place in segmental systems. The simple postulation of a set of features cannot account for such facts.
> [1974 page 6]

To explain the relative frequency or rarity of certain segments, Kean posits "a hierarchy of features which is derivable from the intrinsic ordering ... of markedness conventions" [1974 page 81]. For example, vowels are usually -anterior, consonants are +anterior. It is therefore highly unusual, or marked, for a vowel to have the feature +anterior. But a vowel also has the feature -consonantal, and consonants the feature +consonantal. Therefore, the feature anterior is correlated with that of consonantal; in the unmarked case, we have the following rule:

unmarked anterior --> +anterior/ +consonantal or -anterior/ -consonantal

Forming the complement of this rule, we obtain the convention for determining what the value of anterior should be if it is marked:

marked anterior --> -anterior/ +consonantal or +anterior/ -consonantal

Or, more schematically,

unmarked feature F_i--> αF_i / X
unmarked feature F_i--> $-\alpha F_i$/ X

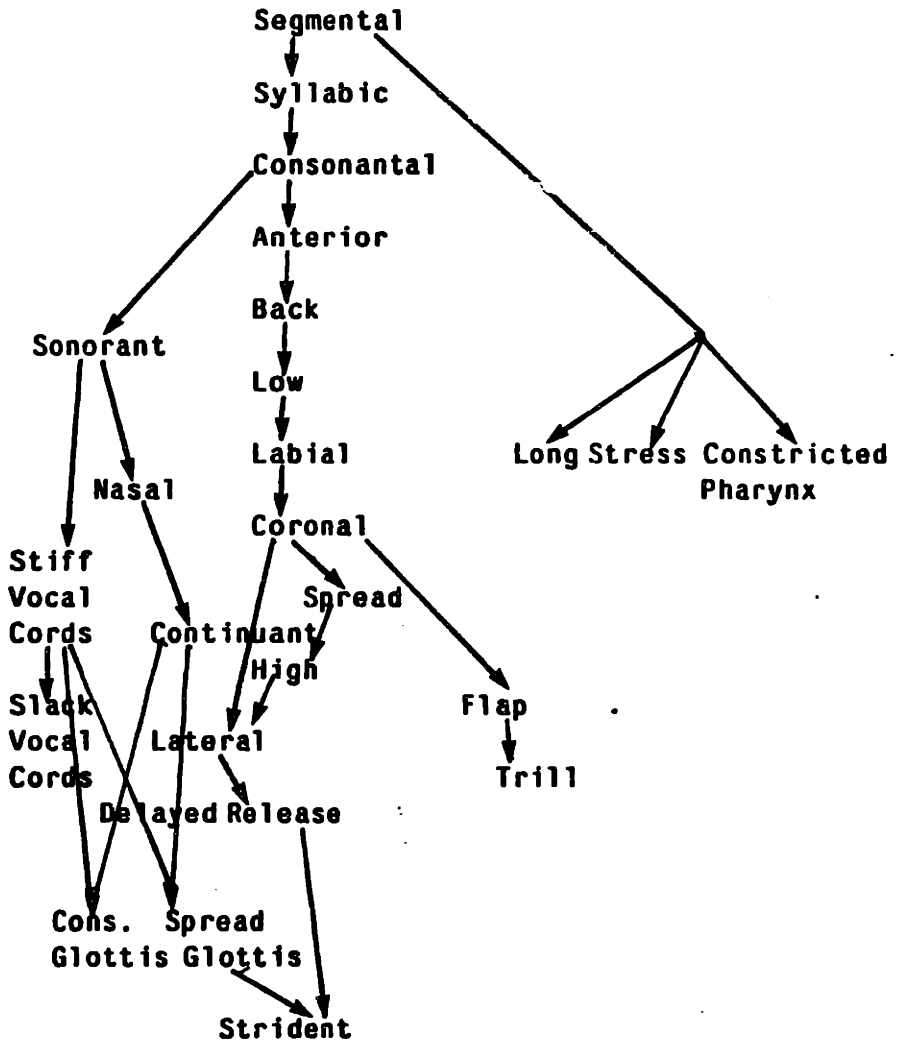
where α = + or -

To determine whether a segment is marked for anterior or not logically demands that the feature consonantal be set first. If the segment is -consonantal, then we would expect the segment to be -anterior (the unmarked case); if +consonantal, the segment will usually be +anterior. Pursuing this approach, Kean goes on to show that whether the feature back is unmarked (expected) or marked (unexpected) is dependent on the value of the distinctive feature anterior. We obtain the following hierarchy of distinctive features:

Consonantal
↓
Anterior
↓
Back

If this analysis is applied to all 24 distinctive features that Kean considers, one arrives at the following

dependency diagram:



Each distinctive feature in the network depends on those features immediately *above* it to determine whether it is marked or not. For example, to determine whether the feature *continuant* is marked or unmarked we must know the values of the the features *coronal* and *nasal*; to know whether *continuant* is marked or not, we must know the values of the features *coronal* and all features above *coronal*, *nasal*, and *sonorant*.

Although Kean did not choose to do so, we may interpret this hierarchical structure as the specification of a developmental program for acquiring a segmental system. According to distinctive

feature theory, segments can be distinguished only if they have different values for at least one of the 24 distinctive features. For example, the segments *a* and *i* are distinguishable given a segmental system that sets all distinctive features for the two segments to the same value save for the feature *back*. *a* is unmarked for back, while *i* is marked for back (i.e., is expected). To take another example, the segment *ae* is also marked for back (*-back*) but is distinguishable from *i* because it is additionally marked *+low*.

We see then that a segment must be explicitly marked in order to distinguish it from the default set of plus and minus values. Otherwise, all segments would be unmarked for all distinctive features, and hence all would possess the same array of distinctive feature values. In other words, if we regard the array of distinctive feature marks as partitioning the universe of possible segments into equivalence classes, then if no segments were marked there would be just one class of segments, the totally unmarked segment.

This remark is not quite accurate, however, since Kean also assumes a basic syllabic/non-syllabic distinction in addition to purely distinctive feature contrasts. As a result, there is always an initial division of all possible segments into two classes: consonants and vowels, according to the rules:[21]

unmarked consonantal--> +consonantal/ -syllabic
marked consonantal--> -consonantal/ +syllabic

Given the initial partition defined by the feature syllabic, we can thus distinguish two classes of segments, even if no other distinctive features are used for marking segments:

(1) {i, e, ae, u, o, oe, i, e, a u , o } (+syllabic)
(2) { p, t, t^y, etc....} (-syllabic)

At this stage then, there are in effect just two segments: "consonants" and "vowels," as defined by local segmental context. To generate new classes, we must mark additional distinctive feature values. The key idea here is that there is a definite order in which new features are used to form new segmental classes. New classes may be formed by splitting established classes, with the split based on the order given by the distinctive feature hierarchy. Suppose we start with a division into just two classes of segments, consonants and vowels. The next partition is based on the next *unused*, (previously unmarked) feature in the hierarchy. This is a natural assumption. We cannot get a new class of segments unless we explicitly mark a distinctive feature contrary to its expected value -- otherwise, we would simply obtain the default feature settings for all later featuresin the feature

21. The initial division into syllabic might be derived from the basic s-w pattern of metrical phonology, in which case the feasture syllabic could be dropped; see, e.g., Vergnaud and Halle [1980]. This matter will not be pursued here.

hierarchy. So we must mark at least one new distinctive feature. Further, since features lower down in the hierarchy depend on the values of features above them, the natural place to look for the next distinctive feature to mark or not is the next feature below *consonantal* in the hierarchy, i.e., either the feature *anterior* or the feature *sonorant*. We cannot skip either of these features to try to mark, say, the feature *labial*, because the value of labial depends upon whether *low* and *labial* were marked or not, and these features have not yet been evaluated. So let us say that, in general, a new partition must be formed by marking exactly one of the distinctive features immediately below the last feature that was marked.

How is a split triggered? This choice must be "data driven" since different segmental systems will have different segments (from the adult point of view) that are marked for a particular distinctive feature. For example, as Kean observes, in Hawaiian only the segment *n* is marked for *sonorant*, but in Wichita, it is *r* that is so marked [1974 page 57]. So a split must be triggered by the existence of some detectable difference between at least one of the members of an existing segmental class and the rest of the members of that class. Presumably, this difference could be detected on a variety of grounds -- articulatory or acoustic minimal pairs; nothing more will be said here about just how this might occur. What one can say, however, is just *where* the next distinction will be made -- the next available unused distinctive feature in the Kean hierarchy. Moreover, the amount of information it takes to split an existing class is also clear -- it will take just one minimal pair distinction, or one bit of information.

As an example, consider again the class of vowels {i, e, ae, eu, ...}. By the hierarchy diagram, the next split of the class must be captured by the value of the next feature below *consonantal*, namely, *anterior*. As it turns out, the feature combination [*-cons +anterior*] is an articulatory impossibility, so that in fact the feature *anterior* cannot be freely varied given that the value of the feature *consonantal* is minus. So the candidate distinctive features that may be used to split the class {i, e, a ...} are the features just below *anterior*, namely, *back* or *sonorant*. The combination [-consonantal, -sonorant] is also impossible, however, so that a potential split must be pursued by considering the two features below *sonorant*, namely, *nasal* or *Stiff Vocal Cords*. For now, let us make the assumption that the feature *back*, being the first immediately available unused feature, is elected to serve as the carrier of the new distinction. Note that in any case features lower down in the Kean hierarchy, e.g. *continuant*, or *strident*, cannot be used at this point to form new classes of segments.

Suppose then that the feature *back* is selected for marking, forming the basis for a new partition of segments. By marking *back* we obtain the following potential classes: *Marked back* {i, e, ae, u...} and *Unmarked back* {a, etc.}. Note that Kean's marking convention, unmarked back--> +back/-anterior, establishes that *marked back* must be *-back* in this case, and *unmarked back*, *+back*. In effect, two "vowels" have been established. corresponding to two possible pathways through the hierarchy diagram. In an actual segmental system, there must be some segment that actually prompts this split.

That is, there must be some segment that is marked for *anterior*, some segment (perhaps the same segment) marked for *back*, etc. This constraint implies that every natural segmental system must have at least one segment that is marked for every feature in the hierarchy diagram. However, this constraint is too strong; as Kean observes, natural segmental systems do not seem to utilize all possible feature contrasts. A restricted version of the constraint does seem to hold, Kean notes. The features in the graph from sonorant and coronal upwards are attested in all natural segmental systems.

How does this process fit into a program size measure of acquisition complexity? Fixing a marked back/unmarked back distinction requires one bit of information. The total amount of external information required to fix a segment corresponds to the number of marks (m's) it receives. Thus, if no new classes were ever formed (hence, no segments ever marked), then no external evidence would be required to fix such a system (and it would have only the completely unmarked segments *t* and *a*). In contrast, a segment such as Φ is marked for three features, *labial*, *continuant*, and *slack glottis*, demanding three externally supplied bits of information to prompt three class splits. In general the acquisition complexity of a particular segment depends upon its hierarchical relationship to other classes that have already been acquired. If the segment is immediately below an established class in the Kean hierarchy, then its specification is largely determined by the specification of the class above it. For example, a segment marked for *back*, *low*, and *labial* is just one mark different from the class established before it that is marked for *back* and *labial*; to specify this new class one need only draw a single new distinction, rather than restate the entire chain of partitions. This information-theoretic redundancy is captured by the hierarchical relationship between the features *back*, *low*, and *labial*. In short, in the case of segmental acquisition one can show that the information-theoretic measure of acquisition complexity and and the linguistic model of complexity coincide. The *size* of the developmental program for a segmental system is precisely the number of decision points that are required to fix the system -- the maximum number of *m*'s used.

Example.

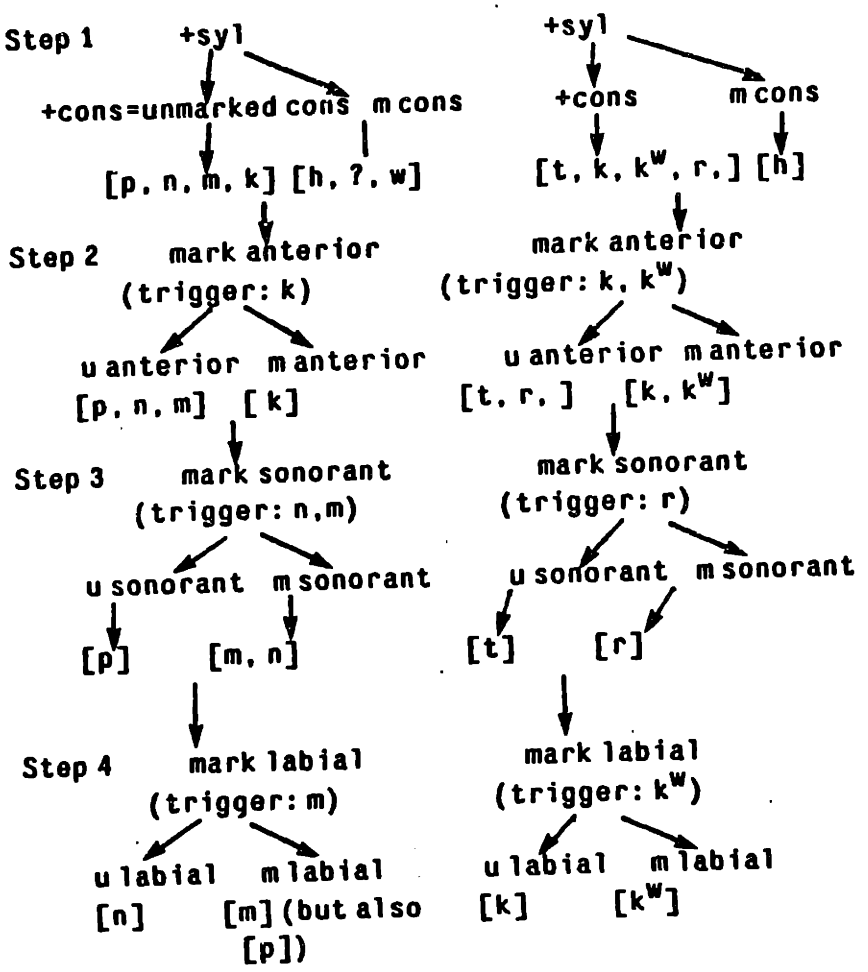
Hawaiian

	p	n	m	k	l	e	u	o	a	i	?	h	w
cons	u	u	u	u	u	u	m	u	u	u	m	m	m
ant	u	u	u	m	u	u	u	u	u	u	u	u	u
back	u	u	u	u	u	m	u	u	u	m	u	u	u
low	u	u	u	u	u	u	u	m	u	u	u	u	m
lab	m	u	m	u	u	u	u	u	u	u	u	u	u
son	u	m	m	u	u	u	u	u	u	u	u	u	u

Wichita

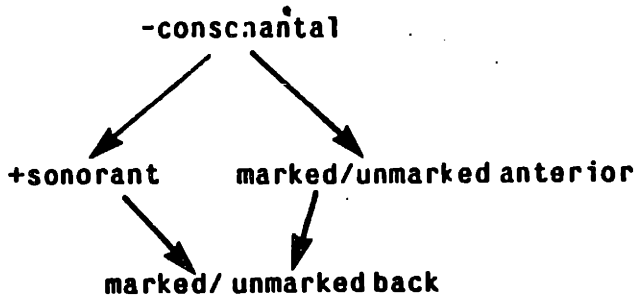
	t	k	k^w	r	h	a	i	u
cons	u	u	u	u	m	u	u	u
ant	u	m	m	u	u	u	u	u
back	u	u	u	u	u	u	u	u
low	u	u	u	u	u	u	u	m
lab	u	u	m	u	u	u	u	u
son	u	u	u	m	u	u	u	u

Hierarchy diagram of example class splits.



Different segmental systems correspond to different patterns of developmental paths. Note that at any given step where a class split occurs one must determine whether the class that is *not* prompting the split is marked or unmarked at step i+1. Thus [p] is marked labial at step 4 even though it doesn't prompt the split into marked/unmarked labial.

The important feature of the partitioning process is that splitting occurs at the leading edge of the directed hierarchy graph, by successive refinement of existing classes of segments:

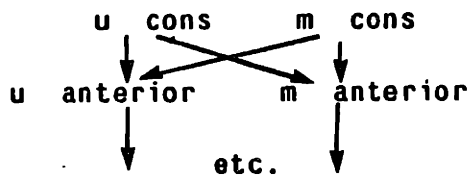


Because extension of classes occurs solely via the refinement of existing partitions, the set of segmental classes at step *i* will be a homomorphic image of all of the other trees before it in the developmental sequence. Also note that the tree is developed by marking just one distinctive feature at a time -- not a necessary constraint, since it is not clear why one could not develop a new class by marking two or more features in one step. The result of this constraint is to guarantee that at any step *i* in the development of a segmental system the classes of segments will be at most one mark (m) different. For instance, this constraint excludes the following array of marks:

	X1	X2	X3	X4
consonantal	m	u	u	u
anterior	m	u	u	u
back	m	u	u	u
low	u	m	u	u
labial	u	u	m	u
sonorant	u	u	m	u

From one point of view the one-mark constraint is a puzzling one. It is not at all obvious why segmental systems should be designed so that the alteration of a single distinctive feature could convert an *a* into an *i*. This would seem to be an unwise design choice from the standpoint of error detection or error correction; as is well known, in order to be able to correct errors of *k* bits, then segments would have to be separated by a ball of radius 2k + 1 (since one must guarantee that changes of up to *k* bits in any two segments still leave one able to determine the original segment). However, as will be discussed in more detail immediately below, the one-mark constraint is a natural one given a model of acquisition from positive-only evidence. Importantly, natural segmental systems seem to obey the one-mark constraint, as Kean observes. If this observation is correct, then the fact that the one-mark constraint fits naturally into a model of incremental, acquisition from positive-only evidence (rather than a model of error detection or correction) argues strongly that the study of acquisition complexity is on the right track.

Finally, because marking is always carried out at the immediate fringes of the existing class tree, the set of possible segmental systems is significantly constrained, and the information load of acquisition correspondingly reduced. Intuitively, this is because the total information content of a segmental system is now encoded via *paths* through the developmental hierarchy, rather than in a matrix of marked and unmarked features. By ordering the marking of segments one can focus attention on just one distinction at a time. Put another way, instead of setting all n features of each segment in a system plus or minus independently, a total of approximately 2^n decisions, one can fix the markings of a complete segmental system in a sequence of *n* decisions over time:



For example, consider the difference between fixing a segment such as *m* in isolation vs. fixing it as part of a larger segmental system. Now, *m* is marked for the features *labial* and *sonorant*. But by the one-mark constraint, there must be some other segment in the system that is just like *m*, except that it is unmarked for *labial*; call this segment X. Segment X (or, rather, the bundle of marked and unmarked features that, so far, identify it as an equivalence class of segments) was identified earlier than *m*. To fix *m* we need only make one additional decision: is there a member of the old class that is marked labial or not? If it is, we obtain an *m* in the segmental system; if not, we obtain an *n* or something close to an an *n* (segment X) but no *m*.

Put another way, the class of possible natural segmental systems seems to be more restricted than one might expect given a model where segments are obtained by selecting arrays of marked and unmarked distinctive features at random. Segmental systems are constrained in just the ways one might expect if they were acquired on the basis of an incremental fixing of segments according to the Kean feature hierarchy.[22]

Let us consider this last point in somewhat more detail. Kean also notes that many segmental systems are not attested, and posits as a descriptive account of this fact what was called above the "one mark" condition.

---

22. Readers familiar with techniques for compact storage of matrices will note that the markedness matrices are for the most part sparsely populated with m's, and those m's that do exist are systematically related to one another. But this is simply to say that there is a more compact representation of the matrix than an explicit list of u's and m's -- namely, as a series of partition decisions.

> (i) For each [m-obligatory] feature there exists at least one segment which is marked for that feature. (These features are numbered i=3 to 8 and comprise consonantal, anterior, back, low, labial, and sonorant.)
> (ii) For each pair of features F_i and F_j, $3 \leq i < j \leq 8$, if there is any segment which is marked for both F_i and F_j, then there are two segments S and S', such that S is marked for one but not both of F_i and F_j, and S and S' agree for all other features F_k, $3 \leq k \leq 8$, $k \neq i, j$.
> [1974 page 61]

This descriptive constraint has two effects. First, it ensures that every natural segmental system has a certain richness; there cannot be a system that has only one or two marks, hence only one or two segments. Second, no segment is two m's different from any other segment, at least for a certain class of primary distinctive features; it can be at most one *m* distinct.

Kean observes that many unusual segmental systems, such as the Hawaiian system presented above, satisfy this constraint. In contrast, the following system violates this condition (and is unattested in natural languages), because *m* is two marks different from any other segment:

Impossible Segmental System

	m	t	k	i	a	u	h
cons	u	u	u	u	u	u	m
ant	u	u	m	u	u	u	u
back	u	u	u	m	u	u	u
low	u	u	u	u	u	m	u
lab	m	u	u	u	u	u	u
son	m	u	u	u	u	u	u

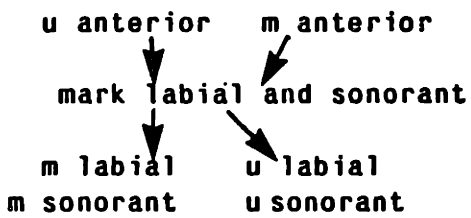
Apparently, as Kean demonstrates, the distinguishability condition is descriptively adequate. But there is no obvious reason *why* this condition should hold. Why couldn't there be a segmental system with segments that were two or more marks different from other segments in the system?

Importantly, the distinguishability condition follows immediately from the markedness hierarchy interpreted as an incremental developmental model. As we have described the way in which the hierarchy can be interpreted as an acquisition model, only one feature can be used to form a new partition of segments -- only one *mark* is ever added at any given step. As a result, at any stage in the acquisition of a segmental system the partitions correspond to segments that are at most one *m* apart, automatically satisfying the distinguishability constraint. So Kean's observation might well be

explained as a side-effect of the acquisition of segmental systems. Even so, it seems as though we have merely replaced a stipulation about the well-formedness of segmental systems with a stipulation about the acquisition of segmental systems. Why should acquisition be incremental?

There are several reasons why one-*m* acquisition steps should be the rule. A weak argument is that it would be natural for the smallest unit of a representational system to be able to function in a model of how changes to that representation can be made; if rules can make reference to the single unit *anterior*, then it is natural to expect that there could be rules to set just the value of the single unit *anterior*.

A stronger argument can be made on grounds of acquisition, as mentioned earlier. Suppose that acquisition is not incremental, so that two or more marks can be added at a single step. It would then be possible to form a new class partition based on marking both the features *labial* and *sonorant* without having first used the feature *sonorant* to form any segment classes:



Now there is no class that will correctly accomodate a segment that is [u labial, m sonorant]. One way to remedy this problem would be to allow the procedure to go back and rebuild classes that have already been formed, but this would violate the developmental ordering that has been assumed. The fringe of the hierarchy tree would no longer summarize the possible next states that could be hypothesized, since there could be segments such as *n* that would demand the interpolation of new classes between older partitions and the current partition. In other words, the one-m constraint amounts to the demand that new classes be the minimally specific refinements of existing classes. This way of ordering hypotheses in a lattice of general to specific descriptions is, as we shall see immediately below, common to several proposed models of so-called "concept acquisition." It is also sufficient to guarantee that the incremental acquisition procedure proposed earlier will work correctly.

Beyond the one-mark constraint, there are apparently other restrictions on segmental systems that may have an explanation in the domain of acquisition, though the connection is not so straightforward. Consider the following array of marked and unmarked [-syllabic, -consononantal] segments as exhibited by Kean [1974 page 47], where blank positions indicate an unmarked segment, *m*=marked:

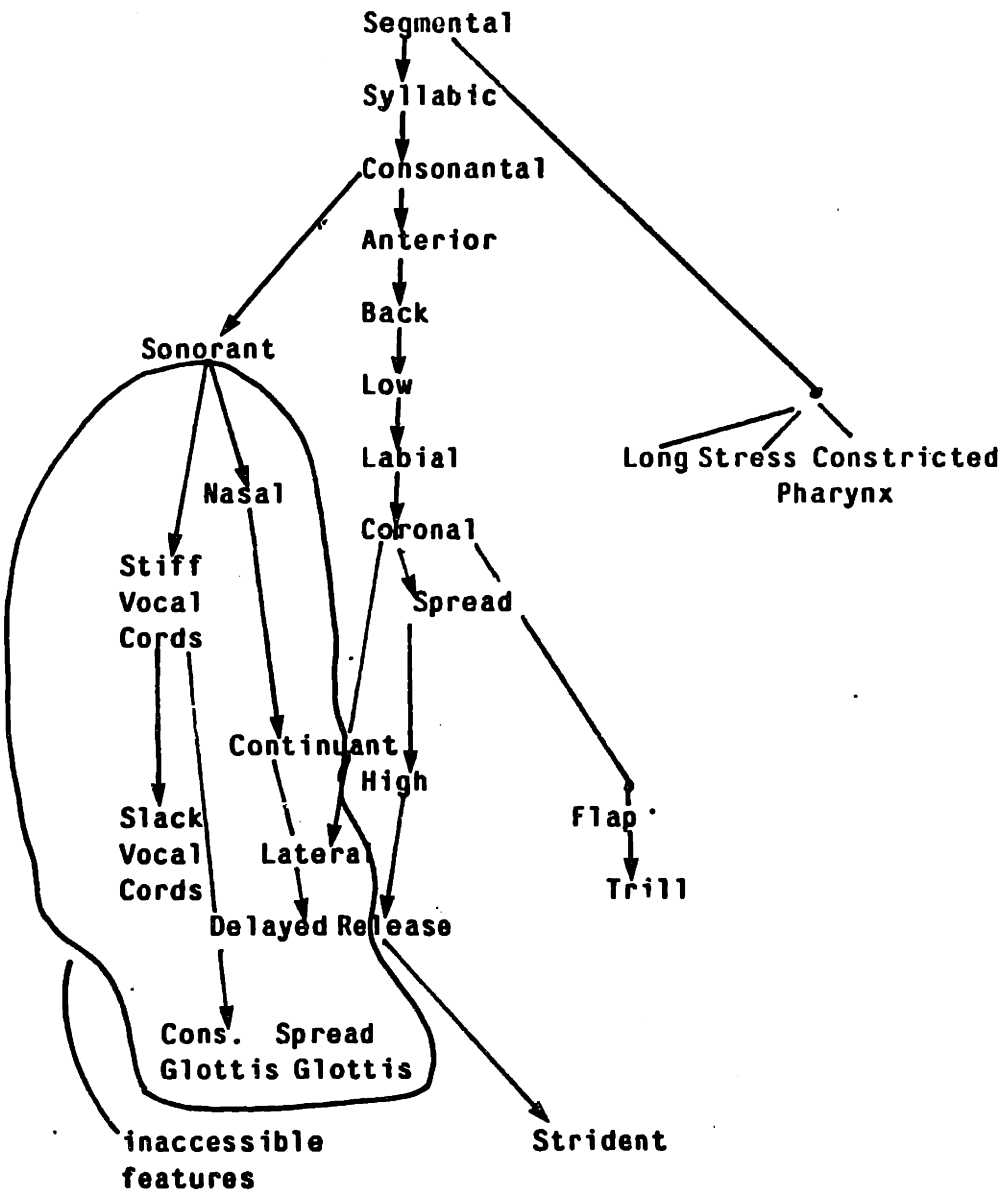
	i	e	ae	u	o	oe	i	e	a	u	o	c
cons												
ant												
back	m	m	m	m	m	m						
low			m			m	m	m		m	m	
cor												
son												
lab				m	m	m	m	m				m
spr												
high		m			m			m			m	
nas												
cont												
StVc												
SlVc												
SG												
CG												
lat												
DR												
stri												
flap												
trill												

While the details of this markedness matrix are not crucial, what is significant is that the matrix exhibits a high degree of redundancy. Apparently, even though there are 2^{24} possible patterns of distinctive feature values -- over 16 million possibilities, as Kean notes -- most of these are not exhibited in natural segmental systems. This sparseness is apparent in the regions of the matrix above that are uniformly unmarked. It seems as though the features *Stiff vocal cord, delayed release*, etc. are not utilized in distinguishing among vowels. As Kean observes, some of these impossible co-occurring feature values can be accounted for since they would otherwise imply the existence of contradictory articulatory features. For example, a segment cannot be both low and high; hence, the value -low implies that *high* must be *+ high*; a segment cannot be both flapped and trilled, hence *+flap--> -trill.* But the great bulk of the redundancy remains unexplained, as Kean remarks:

> Any theory of phonology which assumes the current feature framework must have some mechanism for excluding these nonoccurring segments. The cases [of implicational constraints -- rcb] are meant to be illustrative of the types of segments which must be disallowed in phonological systems. Further research in this area is called for.
> [1974 page 72 n. 6]

The constraint here could well be developmental. Recall that by the rule for forming new classes the features *nasal, Stiff Vocal Cords*, and *back* are all candidates for a partition of the basic vowel class into

two new classes, and that the feature *back* is apparently selected. For whatever reason, the features *nasal* and *Stiff vocal cords* are not used. Without accounting for this last fact, observe that by the definition of the developmental process, we cannot mark a feature (hence use it to create a new partition) unless the features above it in the markedness hierarchy have previously been used to create a new partition. But then, if *nasal* and *Stiff vocal cords* are never marked, then none of the features below them in the hierarchy ever become available for forming partitions of the class of vowels (or, rather, the [-syllabic -consonantal] segments). Given the developmental model, the features *Slack vocal cords, constricted glottis, Spread glottis, strident, continuant, lateral,* and *delayed release* will all remain unmarked, as a body. This is precisely what is observed. Schematically, a whole section of the hierarchical tree becomes accessible because of the "blocking" effect of the *u*'s positioned at the features *Stiff V.C.* and *nasal*, as indicated in the figure below:



It is claimed then that the systematic non-use of certain clusters of distinctive features is a by-product of the incremental acquisition model itself, and in that sense is explained by a developmental model. By restricting the way in which new partitions can be formed out of old ones, the model implies that once certain pathways are not selected for forming segmental classes, they will never be selected. This adds considerable constraint to the acquisition process itself. For instance, once a segment is known as [-syllabic -consonantal], only the following additional distinctions need be made: back, low, labial, coronal, nasal, Stiff v. c., spread, high, for a total of 8. In contrast, if one cannot eliminate such

pathways, then 19 features must be set.[23] The developmental program allow npress" the table of u's and m's exhibited above into a ordered series of 8 step-by-step deci ch new class partitioning depends on the partition decisions that have gone before -- as reflected in the Kean hierarchy. The current state of the class partitions reflects previous decisions; at each step, only a single new markedness decision (determining which distinctive feature should be marked "m") is made. By arranging the decisions sequentially over time yet in correspondence with the hierarchy graph, we reduce the problem of fixing a [-syllabic -consonantal] segmental system from one of choosing a single system out of 2^{19} possible systems to one of making just eight decisions. The developmental model proposed here effects a considerable economy over an instantaneous model that would simply "project" an entire segmental system in a single step given a representative body of data for that system. [24]

The compression in the table of u's and m's for [-syllabic -consonantal] segments is made possible at the price of ordering the decisions about how distinctive features are to be marked. As a result, the model predicts that there should be observable *stages* in the development of segmental systems, corresponding to the implicational structure of the markedness hierarchy. Recall that in Section 1 it was claimed that developmental stages are in fact simply the visible surface residue of such implicational structure. If this is so, then the developmental model just advanced might be expected to imply certain patterns of surface ontogenesis. Segmental classes ought to develop in line with the Kean markedness hierarchy: that is, the first segmental classes formed should reflect the -consonantal/+consonantal distinction; the next, anterior, and so forth.

This proposal is quite similar in spirit to the well-known theory of the acquisition of distinctive feature contrasts advanced by Jakobson (Jakobson and Halle [1956]), though it differs in detail:

> Ordinarily child language begins, and the aphasic dissolution of language preceding its complete loss ends, with what psychopathologists have termed the "labial stage." In this phase, speakers are capable of only one type of utterance, which is usually transcribed as /pa/. ... The development of the oral resonance features in child language presents a whole chain of successive acquisitions interlinked by laws of implication.
> [1956 pages 37, 41]

---

23. There are some complications to this picture that have not been directly adressed here. Some of the impossible alternatives are the result of impossible articulatory co-occurrences, as Kean observes. These result in a series of implicational demands, such as, -consonantal–>+anterior. Because of these implications the account of which features are accessible or not must be complicated. If a feature is set by implication, then features below it in the hierarchy can become accessible, just as if that feature had been used for marking.

24. One could, of course, *simulate* a developmental model in an instantaneous framework, since one is free to specify any function whatsoever to project the initial data into a final segmental system. But this approach would seem to miss the point.

> ...the development of the spirants presupposes that of the stops in children's speech, and in the world's languages the former cannot exists without the latter... The development of the back consonants presupposes in the speech development of the child the development of the front consonants, i.e. the labials and dentals...
> trans. Olmsted [1971], p.106

In Jakobson's theory, as in the model presented above, the development of phonological contrasts proceeds by homomorphic refinement of existing contrasts. It is cruder than Kean's theory because it does not incorporate the articulatory implications of Kean's hierarchy (such as -cons-->+anterior) or the bifurcations of Kean's hierarchy.

Are the predictions implied by the developmental model borne out? The evidence here is difficult to assess, since there are several factors that intervene to complicate the order in which segments appear in child speech. (1) Just because a partition is established, it might not be used, or be necessary to use it; this could depend upon extra-linguistic factors. (2) certain segments might be easier to produce than others. Thus, although *t* is completely unmarked in Kean's hierarchy, and thus should be the first to be produced, it could well be that *t* presents articulatory difficulties, and that *p* is first produced.

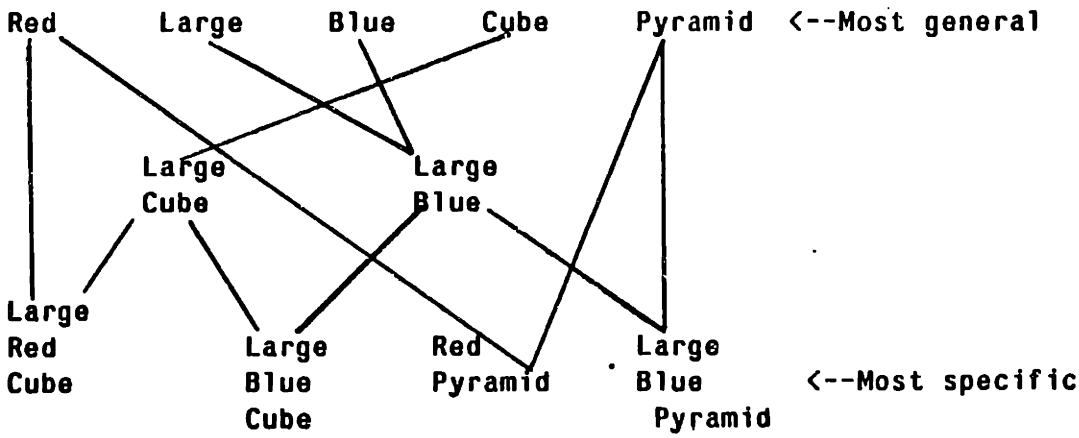
3.2.1.1 Version spaces and the Kean hierarchy

To summarize the discussion so far, we have seen that the order implicit in the setting of marked and unmarked values for the distinctive features of segments can be naturally construed as an acquisition model, and in such a way that one can account for some of the observed restrictions on natural segmental systems. Interestingly, this model is also a variant of a quite general model of "concept learning" proposed by researchers in Artificial Intelligence, the "version space" model of acquisition (Plotkin [1970]; Mitchell [1978]).

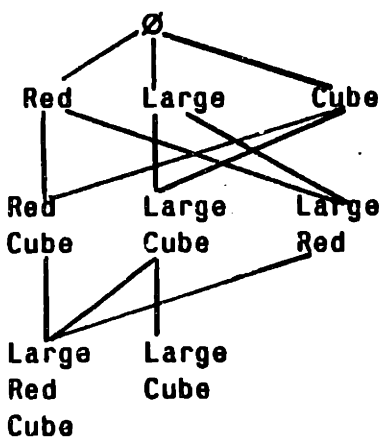
What is a version space? Informally, it is just a compact notation for representing an entire space of possible descriptions consistent with some set of (positive and negative) examples. A description is just an expression in a representation language for the domain of objects under consideration.
-- most commonly, simply a list of *primitive features*. Crucially the description is formed without the use of disjunction. For example, if the domain of examples to represent consisted of simple toy blocks of various shapes, and the description language consisted of the (color) primitives *red, green, blue* and the (shape) primitives *cube, pyramid, sphere*, then the expressions *red*$\wedge$*sphere* or *red*$\wedge$*sphere*$\wedge$*blue* would be valid descriptions. Given a description language, a version space is an acyclic, directed graph of nodes that are expressions in the description language. (Thus a typical node in the simple language of blocks might be *red*$\wedge$ *pyramid*.) Given two nodes n_1 and n_2 in the graph, there is a link from n_1 to n_2 just in case the description denoted by n_1 subsumes that denoted by n_2, and n_1 is the most specific such subsumer. (Formally, n_1 subsumes n_2 if n_2 entails n_1, and n_1 is the most specific subsumer of n_2

iff there is no n_3 such that n_1 subsumes n_3 and n_3 subsumes n_2.) For example, if n_1 = blue∧pyramid and n_2 = blue∧pyramid∧ large, and n_3 = blue, then n_1 subsumes n_2, and is the most specific subsumer; n_3 also subsumes n_2, but n_1 is more specific than n_3.

The effect of the subsumption relationship is to impose an order on expressions in the description language such that the version space graph is a partial order (is transitive, reflexive, and antisymmetric); this is established in Mitchell [1978]. The graph is ordered from most specific descriptions, at the bottom, to most general descriptions, at the top; it is a partial order since two nodes need not be related by any common feature. For instance, a space of examples in the blocks world might look like the following:



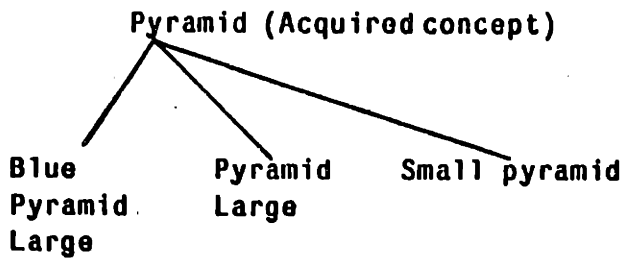
We can (perhaps artificially) bound the graph from above by including the null set of features as the "most general" possible description:



This lattice structure serves as the basic search space for a procedure to acquire the description of a "concept" based on presented examples (and perhaps non-examples) of that concept, as proposed by Mitchell [1978]. The key idea is that the space of possible hypotheses is exhausted by the version space representation. To pinpoint the description of a concept, one need not store the entire space, however. Only two boundary sets need to be explicitly maintained: one the "fringe" of the graph representing the currently most specific description compatible with all the examples seen so far (denoted "S", for "specific"), and the other (actually used only if negative examples are available), consists of the most general description compatible with the examples seen so far (denoted "G," for general). Note that all the descriptions in S will be more specific than those in G. The search for the correct description of a concept proceeds by narrowing G (making it more specific), and broadening S (making it less specific). When and if S and G coincide, the description is uniquely identified.

Turning to the case of the identification of phonological segments, one finds that some of the details of this model are inapplicable. One does not know the marked/unmarked specification of segments; that is the goal of the identification procedure. Rather, one starts with boundary set G = -syllabic or +syllabic, as appropriate, and identifies a segment by moving the boundary of G lower (making G more specific) until all possible marks have been set. Note that segments different from the one being identified serve as negative examples in this case. In both models there is an an incremental, breadth-first search through the sapce of possible feature combinations; at each stage, only the fringe of possible descriptions is retained.

What guarantees are necessary for this procedure to work? The crucial property is that every chain in the graph must have limit points at both the lower and upper ends, as observed by Plotkin [1970] and Mitchell [1978]. Otherwise, the procedure of simultaneous refinement and generalization need not ever halt, since progressively finer distinctions or incremental generalizations can be made, without limit. (Consider approximating an irrational number by a series of rational examples.) Conversely, if every chain in the graph has maximum and minimum accumulation points (either because it is finite or because it is closed in the topological sense), then Mitchell [1978] shows that the incremental acquisition procedure will work correctly. The partial ordering of distinctive features clearly satifies this constraint, since there are at most a finite number of distinctions that can made, with (non unique) lower bounds consisting of all possible distinctions and a (unique) upper bound consisting of the syllabic distinction. Second, the partial ordering must respect the minimal distinguishability condition: a node directly connected to another node must be minimally more general than that node. If not, then the search procedure is not guaranteed to correctly identify a concept description, as the following example shows.



Target: Large pyramid.

In sum, the model of segmental acquisition presented earlier is a variant of the version space model of acquisition.[25]

3.2.1.2 Natural classes, disjunction, and the developmental hierarchy

Kean noted that the hierarchy of distinctive features not only provides a blueprint for a developmental model of segmental systems, it also defines natural classes of distinctive features. For instance, Kean observed that there are no phonological rules of the following form:

-sonorant--> [Delayed release (DR) Constricted Glottis (CG)]
+consonantal--> [continuant labial]

The question is, Why not? There are other attested phonological rules that mention the features DR or labial, and other rules that change more than two distinctive features. The answer Kean proposed was that it seems as if there are no phonological rules that mention features that are not hierarchically related, according to the developmental program. DR and CG do not form a "natural class" of features, because they lie on different branches of the hierarchy tree; in contrast, the feature *labial* lies directly above *high*, and so there can be a rule that combines these two features.

This constraint makes sense from an information theoretic viewpoint, and actually ties together three major themes of this chapter, the definition of a natural class, disjunction, and acquisition from positive-only evidence.

---

25. It should also be pointed out here that the acquisition procedure described in Chapter 2 can be easily modified to reflect the version space acquisition model. In particular, the role of the version space "G" boundary set is played by the rules required to parse sentences, PAR; this bounds the class of sentences handled from above. The "S" boundary set is just the class of sentences producible by the rules of the parser, with the proviso that this class is a subset of P. The producible sentences bound the system's knowledge from below, just as S does in the version space model. When PAR and PRO meet, then the grammatical knowledge is uniquely identified. Note that in this model the system will be able to parse more sentences than it can produce -- as is true, it appears, of the natural case. The details of this approach will not be pursued here.

Consider first how much information it takes to define some class of features. Suppose that the features are not hierarchically related according to the Kean diagram, as in the pair, [CG DR]. To specify the value of DR we must give values for all the features above it in the hierarchy: consonantal, anterior, back, low, labial, coronal, spread, sonorant, high, nasal, continuant, lateral; for CG we must give values for consonantal, anterior, sonorant, stiff vocal cords, nasal, and continuant. Thus the specification of the pair, [CG DR] requires one to provide enough information to write down the entire hierarchy tree, save for the right-hand section that pertains to the features flap, trill, etc. -- a total of 13 distinct features. This is larger than the number of features it takes to specify either CG or DR separately, and thus violates the definition of a natural class, repeated below:[26]

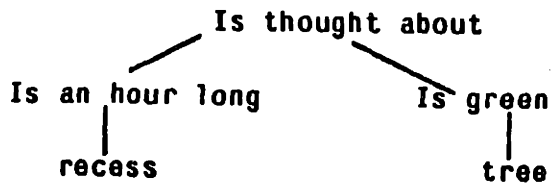
> N is a natural class if it takes the same or less information to specify N as any individual member of N.

In contrast, suppose that two features are hierarchically related, as in [anterior back]. Then the specification of back will almost entirely overlap that of anterior; once we have specified the value of anterior, it takes but one more decision to fix the value for back. Thus the specification of the class [anterior back] will take at most the same amount of information as that needed to specify [back], and forms a natural class. In general, it is easy to see that any pair of hierarchically related features will possess this compression property, and that any pair of non-hierarchically related features will not. It is, of course, the fact that the specifications for anterior and back are redundant, or overlap, that allows us to collapse the description of the pair, [anterior back].

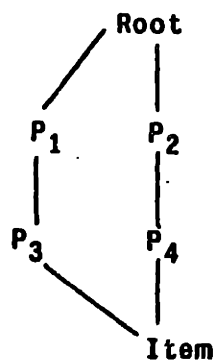
So far, this account of acquisition has been limited to the ontology of *phonological* categories. But there is reason to believe that the model is more general than this. For one thing, as we have seen, there are general models of concept acquisition that incorporate the major features of the phonological case. Moreover, there is another domain of evidence, human conceptual development, that provides support for the model. This is Keil's work on the ontology of concepts [1979]. Keil claims that if one arranges a person's judgments of whether a set of predications of terms "makes sense" or not, then one obtains a characteristically hierarchical tree. By "predication" here Keil means such things as *is loved, is an hour long, think about, is green*, and so forth. Thus, a tree can be green, but cannot be an hour long; a recess can be an hour long, but not green; and both can be thought about. As Keil suggests, following Sommers [1971], we can represent these facts by placing predicates at the nodes of a graph in such a way that an item is dominated by a predicate node just in case it can be sensibly predicated about by that predicate.

---

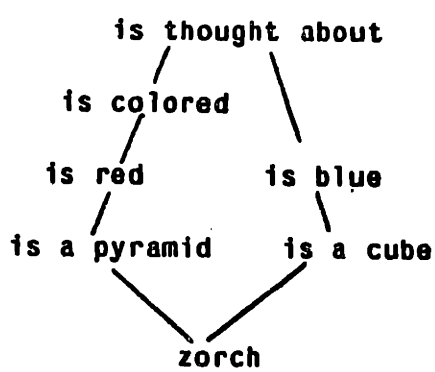
26. This definition is modified from Halle's – it is clear that we have a short description even if a class takes the same amount of information to specify as any individual member, since we want this to hold true if the class has just a single member.



Crucially, one rarely finds that the resulting graph structure forms "M" or "W" shaped patterns -- that is , a case where a single term is subsumed by predicates from two separate hierarchical trees:



For example, suppose that a "zorch" was a word denoting either a blue pyramid or a red cube. According to the M-constraint, "zorch" could not stand for a natural concept; at least not in the vocabulary of blocks used earlier. This is because "zorch" would fall under two separate hierarchy trees, violating the M-constraint:

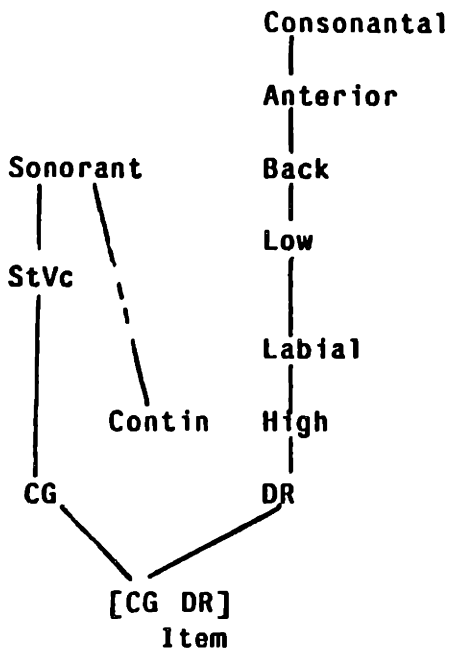


One can see immediately that the M-constraint embodies the same information-theoretic definition of what counts as a natural class as that advanced by the phonological model. "Zorch" is not a natural

kind term because its specification according to the predicate hierarchy requires one to describe the entire predicate tree -- each branch must be listed. But given that this much information could used to pick out the term zorch, then we could use the same capability to specify almost any pair of terminal items in the predicate tree, even if they were not covered by a single label such as "zorch." It is this property that distinguishes a natural from a non-natural concept: if the information required to specify a term is such that a *random* class of items could be specified, then that term is non-natural. This definition is a familiar one in the domain of linguistics, corresponding to the usual intuitive distinction between an explanatory and a non-explanatory account. Roughly speaking, a theory is *explanatory* if it says why things are the way they are rather than some other way (why natural languages look the way they do); a theory is non-explanatory if it could just as easily be set up to predict things contrary to the way they are actually observed. Non-naturalness corresponds to the lack of explanatory power, because so much information is required to pick out a non-natural class that one could just as easily pick out a random collection of items that are connected in no apparent causal fashion; the descriptions of non-natural classes are thus inherently non-explanatory. Chapter 4 pursues this distinction in some depth, providing a formal account of explanatory power and an application to current linguistic theory.

Returning to the discussion of the M-constraint, it is easy to see that there is a close connection between the developmental hierarchy for phonological segments and the predicability trees discussed by Keil. Observe first that the connections among predicates established by sensibility judgements form a subsumption relationship, where a predicate P_1 subsumes entities X, Y just in case one can meaningfully predicate P_1 of X and Y. Thus if P_1=alive, X= tree, Y= fish, Z= rock, then P_1 subsumes X and Y, but not Z. The subsumption relationship is the relationship of "domination" in the tree of possible predications. If there is some predicate P_2s such that P_2 applies to X but not Y, then P_2 dominates X, but not Y. But then, the domain of objects to which P_2 can be sensibly applied is a subset of those to which P_1 can be sensibly applied. We may take this subset relationship as establishing a partial ordering of predicates, just as in the version space or the phonological acquisition model.

Given this partial ordering of predicates, one can now see that M-constraint violations correspond to non-hierarchically related, hence unnatural classes. For instance, consider the class [DR CG]. Represented in the distinctive feature partial ordering, we obtain an M-shaped pattern:

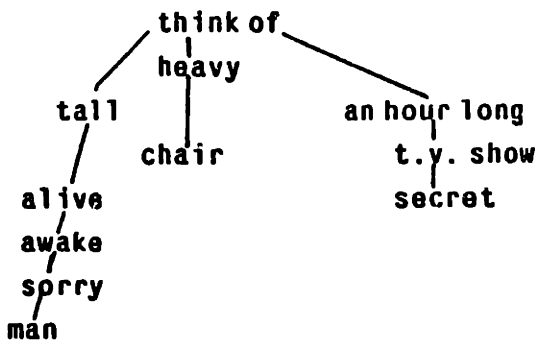


More generally, one has the following result: *a class X creates an M-constraint violation with respect to a predicate hierarchy T iff X is unnatural in T*, where "unnatural in T" can be defined information theoretically, as above, or descriptively, as a term that is defined by non-hierarchically related predicates. Compare Keil [1979 page 161]

> For a concept to be natural it must be composed of predicates that denote the same category or categories that are supersets or subsets of one another.

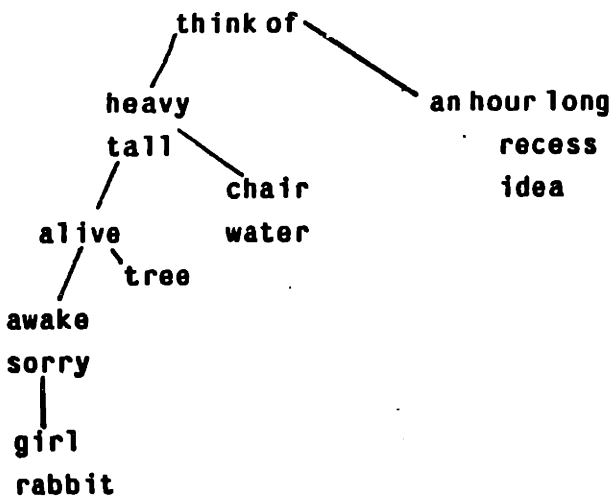
Beyond this connection between the M-constraint and natural classes defined information theoretically, there are other correspondences between Keil's results and the general developmental model presented earlier. Let us review these briefly.

(1) Keil found that predicate hierarchies (predicability trees, as he called them) developed via the refinement of existing classes of predicates, rather than by the radical reconstruction of predicate applicability relationships. This finding was supported by studies of the growth of predicability trees of young children. In other words, Keil found that predicability trees developed via the branching of the fringes of existing predicates -- the same process of homomorphic refinement as was posited in the phonological model presented above. For instance, at the earliest ages studied (5-6 yrs.), some children's predicability trees looked like this:



(This is a single individual's tree.]

When second graders were tested, their trees were foliated versions of initial trees of this kind:



Keil summarizes his findings this way:

> ...children do not suddenly realize that there are physical objects, events, and abstract objects. Rather, they suddenly realize that some things are *not* physical objects. [*Ibid.* 1979]

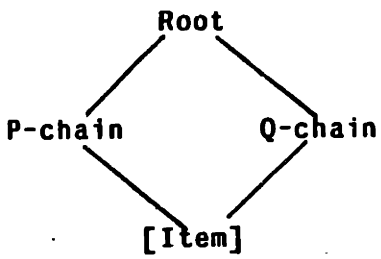
This developmental assumption is just like that of the phonological model. A new phonological partition is formed when it is realized that a segment is *not* like the segments of its classmates, and hence must be marked. (The hierarchy tree dictates *how* it must be marked.)

(2) With rare exceptions, children were found to obey the M-constraint. This constraint is also honored by phonological rules.

(3) The lattice of predications had an upper bound, an all-embracing predicate that could be

applied to *any* object -- e.g., "think about". The corresponding predicate in the phonological domain is "syllabic" or, perhaps, "segmental".

One of the crucial effects of the M-constraint is to eliminate the possibility of disjunction in formulating descriptions of concepts. For suppose that an item is described by predicates that are not hierarchically related (hence an item that constitutes an M-constraint violation), say by the chains $P_1,...,P_m$ and $Q_1,..., Q_n$. Then that item falls under $[P_1,...,P_m] \vee [Q_1,...,Q_n]$, a disjunction that has no compact re-formulation (assuming now a fixed stock of predicates). (The item could be trivially described by the general subsumer predicate "is thought about," just as the null set serves to bound from above all feature descriptions held in commmon, but this description would also admit many non-members of the class.) Thus M-constraint violations must be expressed as disjunctions in the description language. Conversely, suppose that one has a description expressed disjunctively that is known to have no compact reformulation without the use of disjunctions. Then the description is of the form $P \vee Q$, where P and Q are themselves hierarchical predicate chains. But then the description of the item must look like the following:



That is, the description violates the M-constraint. We may conclude that the M-constraint guarantees natural class descriptions of objects, hence compact representations. Further, since M-constrained predicability trees prohibit disjunction, one of the major conditions for the applicability of the version

space acquisition model is met.[27]

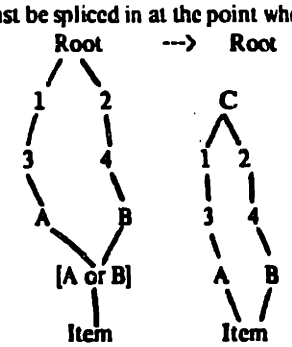
Disjunction and positive-only acquisition.

A major and long recognized problem with the use of disjunctions in description languages is that the danger of over-generalization becomes more acute. By definition, the conjuncts of a disjunctive description have no features in common (save for some very general root predicate). This fact is the bane of simple generalization procedures. For if we attempt to establish one half of the disjunctive clause by observing a positive example, and the other half by another example, then a simple-minded generalizer will often conclude that the correct description of the examples is the intersection of these individual descriptions, namely, the null set. For example, if the correct description of an object is P∨Q, Where *P* and *Q* are themselves conjunctive expressions, then one positive example can be *P*, and another *Q*; the intersection of the two can be empty. Disjunction often results in overgeneralization then, as the predicability analyses make plain. Besides this defect, disjunction also allows non-explanatory, extensional descriptions of data; one can accodomate evidence simply by listing it, in effect.

There are two basic solutions to the problem of disjunction. One is to admit negative evidence, so that one can know if a description is too general. That is the function of the G boundary set in the version space model of acquisition: it delimits the outer boundary of possible descriptions compatible with examples seen so far by describing the properties of negative examples. But what if negative examples are not available? The only other possibility is to eliminate disjunctive statements. How much

---

27. The relationship between lack of disjunction and compact representations raises the question as to whether it is possible to reformulate one's primitive vocabulary and arrive at a new system of predicates in which the formerly disjunctive description is eliminated. In a trivial sense this is possible: a disjunction (A∨B) can always be labelled by its own "name", e.g., C. This certainly leads to shorter descriptions. An approach of this kind has been adopated by some AI researchers, notably, Langley [1977] as a heuristic for discovering new primitives. If one finds a recurring disjunction, then label it with a unique name. Note that if the M-constraint is to be honored then the place at which the new "predicate" C is to be placed in the predicability tree is clear; C must be spliced in at the point where A and B split.



Note that this method, if allowed, would amount to a violation of the incremental, monotonic, approach to acquisition advanced earlier. (The backtracking of the new category heuristic is much like that required to violate the one-m condition for segmental systems.)

disjunction is too much? The following example shows that if "arbitrary" disjunction is allowed then there exists an infinite collection of languages whose individual members cannot be identified from positive-only evidence.

Example. (Angluin, [1978])
Suppose that a family of languages $\mathcal{L}$ includes the "universal" language Σ^*. In addition, suppose that $\mathcal{L}$ contains languages that are defined by rule systems that can express arbitrary disjunctions. That is, suppose that grammars for these languages can be of the form $A \rightarrow \alpha|\beta|....|\zeta$, with no restrictions on the right-hand sides α, β (the rewriting system is still finite in size, of course). The intent is that each part of the right-hand side of the grammar covers one sentence in L_i. In other words, we assume the power to form the language of any member of this family as the union of individual sentences of that language. Let us call these languages the *disjunctive covering languages* of $\mathcal{L}$.

From the standpoint of acquisition, the problem with this family of languages is that any finite collection of samples from a language in $\mathcal{L}$ is covered by two languages: the universal language and the language generated by the grammar, $S \rightarrow I_{(s_1)} \vee I_{s_2)} \vee ... \vee I_{(s_n)}$. But then there is a sequence of positive examples drawn from a member of L that will cause an acquisition procedure to change its guess about the identification of L_i an infinite number of times, and thus the procedure will fail to identify the members of $\mathcal{L}$ in the limit. The sequence is simply members drawn from the universal language. At each step i, an acquisition procedure must guess that examples it has seen have been drawn from either the universal language or the disjunctive covering languages of the examples it has seen. Suppose the procedure guesses the universal language at point i; then no examples it sees later will force it to change its guess, since no example will contradict the guess it has just made. There is some disjunctive cover language whose examples are just like those of the sequence seen up to point i: simply conjoin the examples seen so far, disjunctively; call this language DL_i. But then the procedure must fail to identify DL_i, since there is the sequence of data consisting of the sequence that forced the procedure to guess the universal language followed by examples drawn just from DL_i on which the procedure does not identify DL_i in the limit. Here we make use of disjunctive power. Conversely, if the acquisition procedure guesses DL_i at point i, one can simply add an example that is in UL-DL_i; here we make use of the density of UL. Since we have assumed a countably infinite number of DL_i, the procedure can be made to change its guess an infinite number of times. (This example is a generalization of Gold's [1967] result that any family containing a all the finite languages and one infinite language is not identifiable in the limit from positive-only evidence.)

In the version space framework, the non-identifiability of this family of languages is revealed by the structure of the lattice representing possible identification hypotheses. Since we are using only positive evidence, we cannot move the boundary set G downwards; only S can be moved upwards. No matter what possible hypothesis is made at any step i, the acquisition procedure can be force to make a more refined guess at some later step, moving it to a broader class of languages, simply by

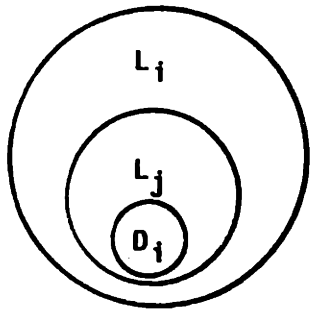
adding an example that was not in the language guessed so far. But since the upper bound G cannot be moved, we can only approach G by moving the boundary set S upwards, one step at a time. Each move represents a new guess; since there are a countably infinite number of moves between S and G initially, the acquisition procedure must make an infinite number of guesses, violating identification in the limit. The crux of the problem is that the space of hypotheses is "dense" in the sense that there are collections of positive evidence compatible with an infinite number of possible guesses. Note that this is the reason that the example family of languages provided by Gold [1967], the family of all languages of finite cardinarily and one language of infinite cardinality, fails to be identifiable from positive-only evidence: there is a finite collection of evidence sentences that is covered by an infinite number of languages in the family.

In short, there are two general situations in which an incremental, conservative approach to acquisition will work:

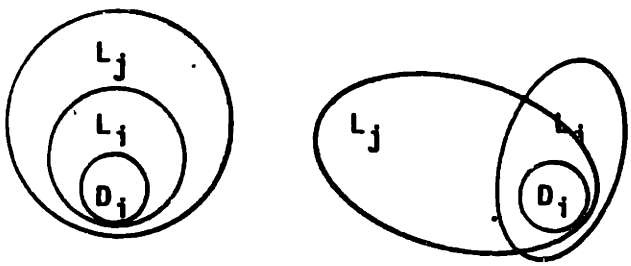
> (1) If positive and negative examples are available, then a partial ordering of hypotheses such that the accumulation points of the ordering are themselves hypotheses suffices. (This is the version space approach.) (Note that, as Gold [1967] shows, there are languages that are not identifiable from positive and negative example sequences. To see this, observe that a data presentation of a language can be represented by enumerating all strings in Σ^* and associating a "1" with the string if it is in the language, a "0" if it is not. Thus each presentation sequence corresponds to a sequence of 1's and 0's, a binary fraction if a decimal point is placed before the first digit. Then a presentation sequence can be thought of as approximating a number, and the acquisition procedure is in the business of identifying this number. But there is no way to tell whether the string of 0 and 1's is rational or irrational -- one can always be fooled.)
>
> (2) If just positive examples are available, then a sparse covering of evidence sets suffices. (See Angluin [1978]). That is, only a finite number of hypothesizable languages L_i can cover any possible evidence set. If general disjunctive statements are allowed -- braces, in the phrase structure notation -- then this condition cannot, in general, be met. (It might be that some other condition could suffice, however.)

Note that the weak generative capacity of the family of languages does not enter into this assessment, except that the results of (2) depend upon the languages in the family being recursive.

The crucial condition required to ensure identifiability from positive evidence may be stated in another way. The problem with overly-dense families of languages is that, no matter what the positive evidence set, so that data set D_i prompts a guess of language L_i, it is always possible to find another language L_j that is compatible with D_i and can be interposed between D_i and L_i:



Suppose that this never happens, that is, that an L_j can never be so interposed. Then either one of two situations must obtain, $L_j \supseteq L_i$, or L_i and L_j are incomparable, but both cover the data set D_i:



In case 1, either L_j is identical to L_i -- hence cannot prompt a hypothesis change, under the Gold assumptions -- or L_j-L_i is non-empty, and there is some *d* in L_j not in L_i. If L_j is the target language, then this example must eventually appear in the information presentation (again under the Gold assumption of a complete presentation), and an identification procedure will have evidence to change its guess. If L_i is the target language, then the procedure will never be forced to change its guess. In case 2, the same situation obtains. If either L_j or L_i is the correct target language, then some positive example must appear that indicates this possibility -- a sentence not in L_i, but in L_j, or a sentence not in L_j, but in L_i. (This is where the recursiveness of the L_i is used.) Therefore, in either case identification in the limit from positive evidence can proceed successfully. Let us say that when a family of languages meets this condition that it obeys the Subset Condition (or that the family possesses the Subset Property).[28] The point of this condition is to ensure that the acquisition procedure always guesses a subset language if possible, that is, the smallest language that is also compatible with the positive evidence so far encountered. (Note that the sequence of guesses

28. Note that Wexler and Culicover [1980] have suggested that an abstract principle of this sort could be crucial for acquisition.

conforms to an incremental search from more specific to less specific languages if the lattice of languages is ordered by the subset relation. Again, what is crucial to the demonstration is the topological structure of the lattice of hypotheses. The particular details of what these hypotheses denote is irrelevant.)

One can conclude that if the Subset Condition holds, then incremental acquisition can proceed correctly. This condition is met by the the phonological model outlined earlier. Given a datum d_i that prompts a new markedness guess G_1, by the incremental, atomic way in which a new markedness hypothese are formed, it is always the case that there is no possible way that a guess G_2 can be interposed between d_i and G_1. This is because a new guesses are ordered by the number of marks they contain, with lower bound = no marks, and upper bound = all possible marks, and $G_2 \supseteq G_1$ just in case the marks G_2 has is a superset of those that G_1 has; finally, new hypotheses are at most one mark different from older partitions. This incrementality condition is what guarantees identification in the limit.[29]

3.3 The Subset Property

The Subset Property was independently observed and formalized in a recursive-function theoretic framework by Angluin [1978]. In an important theorem, Angluin demonstrated that the Subset Property is necessary and sufficient for identifiability from positive evidence.

Given a family of recursive languages, $\mathcal{L}$. $\mathcal{L}$ is identifiable from positive-only evidence iff there exists an effective procedure which on input i enumerates finite sets $T_1, T_2, \ldots$ such that

(i) $T_i \subseteq L_i$

(ii) For all $j>1$, if $T_i \subseteq L_j$, then L_j is not a proper subset of L_i.

It is easy to see that this condition amounts to the Subset Property, with the additional restriction of effective enumeration of data presentation sets. The intent is for the T_i to be "trigger" sets for hypotheses, and the condition amounts to the claim that such finite triggers must exist for positive identification to work.

Since the Subset Property is a necessary condition for positive-only acquisition, it is not surprising to discover that it subsumes a variety of proposals that have been advanced in the linguistic literature as constraints on the order in which acquisition hypotheses should be made. In fact, it appears as though the Subset Property exhausts what can be said about ordering constraints in acquisition. To support this claim we will review several proposals that have been made regarding the ordering of hypotheses

---

29. Actually, since this system is strictly finite, the Subset Condition does no work here at all.

in acquisition, some in detail. First however, we remark that it might appear as though the Subset Property would be difficult to compute. For recall that in general the determination of whether $L_i \supseteq L_j$ is undecidable, for context-free languages and beyond. Therefore it would seem to be a difficult matter to use the Subset Property, at least in general. This is only an apparent difficulty, however, because acquisition of natural languages is not the general case. For example, suppose we limit the data that is used for acquisition to Degree-2, that is, only sentences with at most two embedded sentences.[30] This is the constraint invoked by Wexler and Culicover. (For a formalization of Degree-2 theory, and further discussion, see Chapter 5.) Now the subset computation is decidable. This is because, without recursion, the only way that a string can be arbitrarily long is if Kleene s ar repetition is admitted -- e.g., rules of the form VP--> V NP PP*. But then, it is a simple matter to write out the possible Degree 2 sentences as those accepted by a finite state transition network, without recursion, so these sentences must form a regular set. Inclusion of regular sets is decidable.

Let us now turn to some examples of how the Subset Property works in the acquisition of natural languages. Some of these cases will underwrite the design decisions of the acquisition procedure outlined in Chapter 2; still others will point out useful extensions to that procedure.

For our first case study, we will examine a problem that has been called Dative Overgeneralization. The facts are these. (See Baker [1979]; [1981]; Pinker [1980]) With certain verbs one can interchange direct and indirect objects, but with others the switch is not permitted:

I told a story to Bill.
I told Bill a story.

But:
I said a funny thing to Bill.
*I said Bill a funny thing.

I reported the crime to the police.
*I reported the police the crime.

I gave a book to Bill.
I gave Bill a book.

I loaned a book to the library.
I loaned the library a book.

But:

---

30. One must of course demonstrate that this restriction on input examples suffices to guarantee correct identification of the target language (in this case, a grammar for the target language).

I donated a book to the library.
*I donated the library a book.

I showed the book to Bill.
I showed Bill the book.

But:
I demonstrated the typewriter to Bill.
*I demonstrated Bill the typewriter.

It is assumed that the acquisition procedure does not have access to the negative (starred) examples, which are never produced. The problem, of course, is how the acquisition procedure can avoid the overgeneralization that a Dative shift is always permissable. Suppose that an example such as, *I gave Bill a book* triggers a rule hypothesis that such a shift is possible. On an *Aspects*-type account, as Baker [1979] shows, this would mean a rule something like the following:

X - V - NP - *to* - NP
1 2 3 4 5
-->
1, 2, 5+3, null, null

But this rule admits all of the starred examples listed above. Call the language generated by the rule system incorporating this rule L_i. If only positive evidence is available, there will be nothing to inform the acquisition procedure that it has overgeneralized; equivalently, there is language L_j that is also compatible with the example, *I gave Bill a book* and yet is a proper subset of L_i. The Subset Property is violated. If positive-only acquisition is to be maintained, then there must be some other triggering sequence of examples that avoids this pitfall.

Baker's [1979] solution to this problem is the obvious one: ensure that an interposed L_j *cannot* be generated. This he does by triggering the rule of Dative shift on a verb-by-verb basis. That is, if we associate with a verb's lexical entry its possible subcategorization frames, then the frame [NP NP] (Dative shift) is associated with a particular verb if and only if a positive example exhibiting that frame has appeared with that verb. Thus, *give* would allow Dative shift as soon as *I gave John a book* was encountered; but since such verbs as *donate*, *demonstrate*, and *say* never appear with the Dative shift frame, the acquisition procedure would never generalize to include them. It should be clear that this "rule restricting" method eliminates the possibility of overgeneralization by positing the most conservative acquisition possible, thus ensuring that the Subset Property obtains.

There is something unsatisfying about this approach, however. While it guarantees by fiat that acquisition by positive-only evidence will work, it leaves unexplained the apparent productivity of

sentence variants like Dative Shift.[31] It also smacks of rote acquisition: if one admits the possibility of rule by rule acquisition with no generalization, then it should also be possible to acquire language on a sentence-by-sentence basis, save perhaps for a rudiment of recursion made possible by simple phrase structure rules.

More importantly, it does not seem to be even descriptively adequate, at least in the simple form that Baker [1979] provides. For, as pointed out in Stowell [1981] even a single verb such as *throw* sometimes permits Dativization and sometimes does not:

> I threw the ball to Bill. -- I threw Bill a ball.
> I threw the ball to the ground -- I threw the ground the ball.

The subcategorization frame of *throw* would have to be annotated in some way to accomodate facts such as these.

Finally, conservative acquisition does not appear to be faithful to the facts about child acquisition, as noted by Pinker [1980]:

> ...it is held that children learn conservatively, entering predicate-construction pairs into their lexicon only upon hearing a particular predicate used in a particular construction.
>
> Unfortunately, this solution will not work. It predicts that children will never generalize a known predicate to a new construction, and this predication is false. Bowerman (1978) documents her young daughter's productive mis-use of verbs in causative constructions, along the line of "I jumped the doll,"... It seems then that children are neither purely conservative (item-by-item) nor purely liberal (lexical redundancy) learners.
> [1980 page 80]

To account for the apparent pattern of under- and over- generalization in child acquisition, Pinker [1980] proposes that productivity can be accomodated by the progressive loosening of "only if" conditions on the application of so-called "lexical rules" that have the effect of Dative Shift, making the Direct Object the Indirect Object, and vice-versa:

> One account for these data is that children's lexical rules first apply only to narrowly defined semantic or thematic categories such as agent-of-action and patient-of-action. The difficulty with this account is in developing a plausible learning mechanism that will take the child from rules that operate on semantic or thematic symbols to the correct adult rules, which operate on grammatical functions [e.g., thematic roles are such things as

---

31. Baker [1979] claims that no productivity has been observed with Dativization in child language. Productivity is observed with other constructions, however, as discussed below.

> Agent, Patient; grammatical functions are roughly the traditional notions of Subject, Object --rebj. It could be that this change simply results from maturation, but I think one would want to implicate nonstationary mechanisms only as a last resort.
>
> There is a better solution. One can invoke a certain property of lexical rules that is independently motivated by studies of adult grammar. In some languages, lexical rules may apply only if certain thematic conditions are first met. For example, a passive lexical rule changing an object to a subject might apply only if the original subject and object were the agent and patient arguments of the predicate (Bresnan, personal communication). It could be that when children coin a lexical rule, the rule operates on grammatical functions, as it does in the adult grammar, but a set of "only if" provisos referring to the thematic relations of the arguments to the predicate is included with the rule... If subsequently a sentence is encountered which inspires the learner to form a rule with the same function replcements, but which simultaneously violates one of the "only if" conditions (e.g. "Leon was considered by Cindy to be a fine yodeler"), that "only if" condition is deleted. Thus the form of the rule remains constant, though its conditions for application might change depending on the particular target language.
> [*Ibid.*, pp. 81-82]

Note that what Pinker is proposing amounts to a claim that the Subset Property holds -- as it must. The rule prompting Dative Shift is to be annotated with feature predicates that hold of semantically defined classes of verbs. A positive example exhibiting Dative shift acts as a trigger by showing that a certain semantically coherent class of verbs can undergo Dative shift -- presumably, not the class that includes verbs such as *say*, *demonstrate*, or *report*. That is, if T_i = a positive example including *I gave Bill a book*, then L_i is to include (perhaps a subset of) that verb class that undergoes Dative Shift. The problem now lies in uncovering the requisite distinctions. Pinker admits that this is a difficult hurdle:

> Alas, one problem remains. There seem to be pairs of semantically similar predicates in English such that only one member of the pair can be operated on by a lexical rule; for example, dativization applied to "give" versus "donate," "tell" versus "say," "show" versus "demonstrate," and so on. Unless someone can discern some subtle semantic/thematic difference between all predicates that can undergo dativization and all those that cannot, the only solution to this acquistion problem is to hypothesize some kind of information indirectly available to the learner that will inform him that a predicate cannot appear as a given lexical form.
> [*Ibid.*, p. 82]

As we shall see below, however, there seems to be no need to draw a *semantic* class distinction to account for these patterns of Dativization. Stowell [1981] observes that the distinction between "show" and "demonstrate" -- "show" can undergo Dativization but "demonstrate" cannot -- is apparently correlated with a Germanic/Latinate distinction in word formation, with word formation being in turn the key to the formation of Dative construction. Though some complications arise in this story -- it is, I believe, linked to deeper properties of residual Case assignment of Latinate forms -- it at least indicates that semantic class distinctions among predicates are *not* crucial here.

Let us now backtrack to reanalyze this same acquisition problem from the standpoint of recent transformational theory, Government-Binding theory [Chomsky 1980]. Contrary to some assertions that have appeared in the literature, (e.g. Baker [1979], Pinker [1980], GB theory actually provides a solution to the problem of Dative Shift acquisition, and in a way that surmounts the difficulties of the lexical approach.

First, one must point out that it is simply an error to assert that a transformational grammar cannot make reference to the properties of lexical items. All modern generative grammars, transformational grammars among them, consist of several modular components. A transformational grammar is not just a set of transformational rules; in addition, there is a lexicon, a set of semantic interpretation rules, and, as outlined in the introduction to this chapter, a set of principles that in conjunction with the X-bar theory define a set of valid base phrase markers. In particular, the properties of lexical items play a central role in Government-Binding theory, in virtue of the Projection Principle. The Projection Principle ensures that the properties of lexical items, including the subcategorization frames of lexical items, are available at every level of linguistic representation. Therefore, these properties may be assumed to be accessible at the point where S-structure (annotated surface structure) is built.

Suppose then a sentence such as, *I gave Bill a book* is encountered. What must be known in order for the sentence to be analyzed successfully? Plainly, there is a crucial prerequisite: the subcategorization frame of *give* must be known, at least in part. As Chomsky observes, to know what *give* means is to know at least that *give* takes an NP (Direct) Object, and an NP (Indirect) Object. Thematic roles must also be assigned correctly. Developmentally, an acquisition procedure could infer at least the presence of the NP direct object from the existence of sentences such as, *John gave a book to Mary*. Note that this means that there is then at least some ordering in the way examples are used for inference. Sentences where Dative shift has *not* applied are assumed to be used in acquisition before those where Dative Shift has applied. How can this ordering be imposed? Recall that we have ruled out the possibility of the external environment supplying this kind of constraint. However, ordering need not necessarily be supplied by the external environment itself, but might, rather, be imposed by the capabilities of the acquisition procedure -- an intrinsic, rather than an extrinsic ordering.

There are several ways this intrinsic ordering could arise. What happens when the procedure attempts to analyze a Dative shift *give* sentence? Assuming that some type of adjacency constraint on NP arguments has already been acquired (see below as to why this would be so), the NP directly after *give* will be taken as the main argument to *give*, i.e., it assumed to function the way *Mary* does in *John*

kissed Mary.[32] But this analysis gives the wrong annotated surface structure for *John gave Bill a book*, since it implies that *Bill* is what is given. Somehow, the acquisition procedure must be able to recognize that this is so. How could an acquisition procedure know this?

In this regard, one might make an assumption similar to that advanced by Wexler and Culicover [1980] in their mathematical study of the acquisition of transformational grammar. Wexler and Culicover assumed that an acquisition procedure could independently construct a correct deep structure for a given input sentence, perhaps from "context":

> The deep structure is the level of structure of a sentence before transformations apply. If much semantic interpretation is related to the deep structure of a sentence, and if a child sometime has available the semantic intrepretation of a sentence even when he doesn't understand its syntax, perhaps it is plausible that in some cases the child might be able to compute what the deep structure of a sentence is even when he doesn't understand the syntax of a surface sentence.
>
> We are proposing that the possibilities of semantic interpretation might sometimes allows the learner to reconstruct the deep structure of a sentence...We assume that the child has cognitive abilities that allow him to do it...it is conceivable...that the possibilities for base grammars are very limited. Therefore, it may be that the central rules of base grammars are learned easily and early. At the point when transformations are being learned, the syntactic rules of the base grammar maybe available to the child. Suppose that the rules of translating between the base grammar and semantic interpretation are avilable to the child... Then if the child has [the] semantic interpretation of some sentences available, he will be able to reconstruct the base phrase-marker.
> [1980 pp. 82-83]

The Wexler-Culicover acquisition procedure determines if its current transformational component is correct by applying its current rules to this known base structure and seeing whether the surface string its transformational component produces matches the input string just received; if not, it knows that its current rules are somehow awry. Similarly, Pinker's [1980] acquisition model for lexical-functional grammar assumes roughly the same input as the Wexler-Culicover approach:

> I assume that the input to the child is a semantic text consisting of strings paired with "uncommitted" f-structures ["functional structures", the lexical-functional theory's analogue of deep structures -- rcb]. These f-structures differ from the ones that the adult would assign to sentences in that obviously they cannot contain information that is determined by rules that have not yet been learned. However, I assume that they contain information about the propositional structure of the sentence's meaning (such as the

---

32. In other words, predicates are assumed to be binary, by default. The existence of additional arguments must be supplied by positive examples. Note that this assumption could be derived from the Subset Property: a rule system where only one (NP) verb argument is permitted generates a narrower class of languages than a rule system that is polyadic. (So actually the true default assumption would be to assume a predicate to be monadic. But this assumption is quickly refuted by positive examples.)

> correct links between predicates and arguments), the topic-comment distinction dfined by the pragmatics of the utterance, and whatever default grammatical information may be associated witht he predicates and arguments (e.g., the assignment of SUBJECT to agent arguments). ... In this example ["John convinced the milkman to swat the dog" -- reb] the child is assumed to know that "John" is the topic of the utterance... that the principal predicate, "convince," takes three arguments, corresponding here to "John," "the milkman," and "the milkman swats the dog," that the predicate "swat" takes two arguments corresponding to "the milkman" and "the dog," that the same milkman is simultaneously the convinced and the swatter, and the the "subject," "object," and "S-complement," respectively, correspond to the agent, patient, and proposition arguments of the predicates.
> [1980, pp. 52-54]

Both models assume then that the canonical argument structure of a verb has been determined before further acquisition proceeds -- Wexler and Culicover, in their assumption that base rules are acquired early, and that correct base structures can be inferred from context, and Pinker in the assumption of an f-structure input that includes correct base structure (namely, the f-structure itself) along with the proper assignment of arguments to predicates.

In the case at hand, we assume that the acquisition procedure knows from "context" that in *John gave Bill a book*, (i) the thematic role of the NP adjacent to the Verb is to serve as the thing given; and (ii) that *Bill* is <u>not</u> the thing given. Note that assumption (i) follows from positive evidence provided by single VP-argument sentences: in *I ate the candy*, *candy* is the thing eaten; similarly for *I ate the cake*, etc. Now we invoke the incrementality assumption adopted in Section 2 and used by the acquisition procedure of Chapter 2, a principle of "inference from clear cases". Basically, the idea here is that no more than <u>one</u> decision is made in a single step. In the case of *John gave Bill a book*, two decisions must be made: one to add to the subcategorization frame for *give*, and one to permit a change to the default correlation between adjacency and thematic role. Since the sentence violates the canonical thematic-syntactic alignment, the acquisition procedure can conclude nothing about the subcategorization frame for *give*. In effect, the sentence is interpreted non-syntactically, and prompts no change in the acquisition procedure's syntactic knowledge (in this case, the subcategorization frame for *give*). In contrast, suppose that the sentence *Bill gave a book to John* was encountered. Putting aside the problem of analyzing the indirect object, note that by the adjacency constraint and the default association between the NP adjacent to the Verb as the thematic role of Patient, the procedure deduces that *a book* is the thing given -- and this is confirmed by an independent reconstruction of thematic roles, or even a partial reconstruction in this case. It is this confirmation that allows the procedure to add an entry to *give*'s subcategorization frame. In the case of *John gave a book to Bill*, only the subcategorization frame of *give* must be altered, and so this is permitted. As a result, the Dative Shift sentence will be ignored as a source of information for syntactic acquisition *until* the unshifted form has been analyzed. Note that this assumption of incremental decision-making resembles the phonological one-mark constraint, and is in fact intended to order the space of

acquisition decisions so that the Subset Property is automatically satisfied.[33]

Alternatively, and more speculatively, one could suppose that the acquisition procedure's abilities at analysis are severely limited, in that at most only a single new grammar rule could be added in the processing of a sentence or a phrase. Then in a sentence such as, *John gave Bill a book*, only *Bill* would be processed, and the mis-match between *Bill* as argument to *give* and actual thematic structure as inferred from context would suffice to tell the acquisition procedure that something had gone wrong. In contrast, the sentence, *John gave a book to Bill* would result in *a book* as the object given. *Bill* would be ignored, but no mis-match would arise. A later sentence would prompt the procedure to tackle the *to*-NP phrase; since a rule to analyze Direct Objects would now exist (perhaps in the form of a partial subcategorization frame for *give*), the procedure could now handle this example. In essence, the subcategorization frame for verbs that take multiple arguments would be built up one argument at a time.[34]

In either case, a full analysis of Dative Shift examples is delayed until a non-shifted example is encountered. Suppose then that the acquisition procedure at last encounters a sentence such as, *John gave a book to Bill.* As before, assume that the thematic roles of NP arguments can be recovered, so

---

33. It is important to understand what incrementality means. Incrementality in underlying acquisition decisions does not necessarily imply incrementality in surface behavior, or rule-by-rule acquisition. This is because a single decision could set a parameter that could have far-reaching effects throughout a grammar, leading to non-continuous changes in surface behavior. For example, once the basic Head-Complement order is set via one example, e.g., Head-Complement order in VPs, then by the X̄ theory it is set for all categorial types. The setting of this parameter is an incremental decision, but it causes more than one surface effect.

It has been assumed here that the adjacency requirement on Direct Objects is the default, unmarked case; for discussion, see below. But in other languages this condition is weakened, even absent. Triggering data for this change could consist of interpolated AP's: V--Adv--NP, where again the NP is confirmed as having the thematic role that an adjacent NP should have. Again there is only a single difference between this example and Adv--V--NP, a minimal pair that focusses attention on the decision to weaken the adjacency requirement. (There is an alternative interpretation of the adjacency requirement, however. A Dative Shift sentence also seems to violate the adjacency constraint. As will be discussed in more detail below, Stowell [1981] argues that this is the result of a word formation rule that "absorbs" the NP Indirect Object into the Verb. If this view is correct, it might be that so-called "free word order" languages are simply languages that are freer in their word formation rules, permitting adjacency even across multiple Noun Phrases. For a similar approach, cf. Hale's [1979] "interpretive" parsing account of W* languages.)

The assumption of thematic-syntactic alignment, more generally the notion of inference from clear cases, has been advanced by other researchers. Compare the following quote from Jakobson [1962]: "At first, child's language is devoid of any hierarchy of linguistic units and obeys the equation: one utterance -- one sentence -- one word -- one morpheme -- one phoneme -- one distinctive feature." Limber [1973] also proposed a model of language acquisition in which syntactic-thematic alignment is assumed as a default, only to be weakened by later examples. More recently, Limber's view has been revived (apparently independently) by Grimshaw [1981].

34. Note that this assumption has the effect of making the acquisition procedure more incremental, thus allowing for ways to guarantee that the Subset Property holds. This extreme version of incrementalism is not adopted by the acquisition procedure outlined in Chapter 2, because it has seemed reasonable to avoid making assumptions about processing limitations of this kind. Some researchers have noted, however (see Newport, Gleitman, and Gleitman, [1977]) that there seems to be a production (output) limitation of a similar sort.

that the acquisition procedure knows that the NP *book* fills the thematic role of the object given. Now the NP is in the same position as the NP arguments of simple transitives, e.g., *John met Mary, I wanted a book*, and so forth. Thus the reconstructed thematic role matches the one deducible from syntactic information, and this alignment permits an entry to be made in the subcategorization frame for *give*:[35]

give: [NP ....]

The NP argument is assumed to be obligatory. This is the right choice if the Subset Property is to hold. For if the NP was assumed to be optional, then two surface forms could be generated, one with the NP argument and one without. Call the language that includes these strings language L_i. However, the correct target language could be one including just obligatory NP arguments for this verb, L_j. Thus there is an L_j that is a proper subset of L_i for this class of triggering data, a violation of the Subset Property.[36] Since the Subset Property is a necessary condition for positive evidence acquisition, the acquisition procedure must be designed so that arguments are assumed obligatory unless specific triggering evidence is encountered that indicates that they may be optional, or absent from a verb's subcategorization frame.

The second portion of *give*'s subcategorization frame critical for the analysis of Dative constructions is the *to*-NP phrase. Once again, there would seem to be plausible available positive evidence for this kind of phrase:

I went to the store.
I took Mary to school.
I walked to school.

Thus we may assume that the acquisition procedure already has the rules to combine *to* and an NP into a Prepositional Phrase-type complement to a Verb. If it does not, then Dative constructions with *to* will simply be unavailable as a source of information for syntactic acquisition, though of course the

---

35. Alternatively, one could assume that the subcategorization frame for *give* is partially built when either a Dative or non-Dative sentence is encountered, in either case yielding the following:

give: [NP, NP]

-- with the *to* preposition ignored, as is apparently often the case in child language. The entry is unrefined, in that none of the distinctions required in the adult grammar are made -- besides the lack of the *to* case marker, the restrictions blocking ungrammatical adult examples are not present.

This second alternative resembles Braine's [1971] proposal, in that a very general form is posited first, only to be replaced by refinements.

36. This argument has also been presented informally by Roeper in Baker and McCarthy [1981], pp. 139-140.

sentences may still be perfectly well *interpreted.*[37] The subcategorization frame for *give* would now look something like this:

give: [NP, *to*-NP]

The order of the NP arguments is not fixed by subcategorization frames. In English at least, this order is set by an adjacency requirement on Case assignment: an NP receiving Objective Case must be adjacent to the Verb that assigns Case:

*I gave quickly a book to Bill.

How is the adjacency requirement acquired? Once again, the Subset Property may be invoked. The most restrictive assumption possible is that adjacency holds, since this generates the *narrowest* class of output possibilities. To assume otherwise would be to guess a language that could be too large, hence a possible Subset violation. A language satisfying the Adjacency condition could be a proper subset of one that was not, and yet cover the same triggering data. The acquisition procedure thus assumes an adjacency requirement as the default, unmarked case, loosening it only if positive examples are encountered that indicate violations of adjacency. In English, since examples violating adjacency (*I hit hardly Bill*) will never be encountered, this strict requirement will never be dropped.[38] In other languages, positive examples exhibiting adjacency violations would prompt a relaxation of these conditions, perhaps along a continuum of possibilities. Thus one might expect to find languages where strict adjacency was relaxed according to a hierarchy of phrasal types. For example, since adverbs and particles are not assigned case, it would seem plausible that the smallest kind of adjacency relaxation would be to allow adverbs and particles to be interposed between a Verb and its Object <examples?> In other languages, such as classical Arabic, a Subject NP can be interposed between Verb and Object in surface forms. The most extreme violations of adjacency would occur in languages that have so-called "free word order," such as Japanese, or Warlpiri. Here, the adjacency requirement is weakened to such an extent that a whole series of NP's can be interposed between the Verb and the

37. There is some evidence that at an early age children in fact ignore *to* PP's; see Maratsos [1978], and the discussion on the acquisition of passive constructions in Chapter 2.

38. Perhaps the requirement is never dropped in any language. Then apparent adjacency violations are just the result of freer use of word incorporation rules. In this case, the Subset Property would still hold -- English would simply have a more tightly constrained set of incorporation rules.

NP receiving Objective case.[39] In fact, as pointed out in the previous footnote, English has a weakened adjacency requirement on Prepositional Phrases of just this kind. On this account, there are no "configurational" or "non-configurational" languages, but simply languages that exhibit a range of adjacency requirements for one or another type of phrase. The retention of the adjacency requirement for NP Objects in English makes it characteristically configurational, because the assignment of thematic roles via the connection of grammatical relations retains its structural properties. (For additional discussion on the configurational-non-configurational distinction and how it might be related to the operations of a parsing device, see Section 4 below.)

Returning now to the main discussion, the acquisition procedure has so far built the basic subcategorization frame for *give*, and has established the (default) adjacency requirement on the NP Direct Object. To proceed further, one must make some assumptions about Dative in the adult grammar of English, within the Government-Binding theory. Here the approach of Stowell [1981] will be adopted. Stowell's theory of Dative Shift is a good model of the change in point of view from *Aspects* to the GB theory, from a theory based largely on a system of *rules* to one based on a system of *principles*. Instead of positing a rule of Dative Shift that is encoded as a transformation, Stowell advances several constraints on Case assignment and how thematic roles are determined so as to obtain the observed array of Dativization judgments.

Stowell assumes that Dativization is possible when two requirements are met, and that the *to*-NP form is possible under a third condition:

> (i) The Verb-NP pair can be reconstituted as a complex Verb. That is, one can restructure $[_{VP}\ [_{V}$ gave$][_{NP}$ Bill$]]_{NP}$ a book]] into, $[_{VP}\ [_{V}$ gave $[_{NP}$ Bill$]]\ [_{NP}$ a book]], thus preserving the adjacency condition on case assignment for the Object NP *a book*.
>
> (ii) The NP can be incorporated only if it can "possess" the direct object of the verb, in some sense. Thus, one has *John gave Bill a book* (because Bill can possess the object, a book), but not *John sent Canada a telegram* (unless Canada is construed as an entity in

---

39. Warlpiri is even "freer" that Japanese in this respect, in that VP's are apparently discontinuous.

It is interesting to observe that if an adjacency restriction is an unmarked, default assumption, then such a constraint could aid the learner even in so-called non-configurational languages where adjacency does not seem to play a central role in the adult grammar. For suppose that the learner assumes that adjacency holds. Then sentences in which the adjacency requirement happens to be met will be interpreted correctly, or at least more easily, than those where it does not. Establishing this case might permit the learner to determine other connections between the Verb and its Object -- namely, case agreement marking. Then, when a non-adjacent example is encountered, the adjacency requirement could be dropped in favor of the agreement marker. Note that this is essentially what has been assumed above about the acquisition of *to*-NP complements. The earliest examples are presumed to be sentences such as *John went to school*, where the constraint of adjacency and the prepositional marker *to* coincide. In English, the adjacency requirement is dropped in favor of case assignment via the Preposition -- hence English Prepositional exhibit relatively free order, modulo the adjacency restrictions of other phrases. In effect, the primitive of string adjacency has been exploited as device to acquire another source of knowledge, that of case agreement.

some way that can possess a telegram, e.g., as an institution.) Note that an animate/inanimate distinction is not the right distinguishing feature here, since *I gave the dock a hard shove* is perfectly fine. Here, the dock can "possess" the quality of being given a hard shove.

(iii) Finally, a *to*-NP form can appear if the verb takes an object that is assigned the thematic role of Goal, or a thematic role with some kind of inherent directionality. Thus, *I sent a telegram to Canada* is grammatical, because *Canada* is an appropriate NP to serve as a Goal.

These restrictions (plus one final twist, described below) do seem to do the trick. For example, as Stowell notes, *That storm almost cost them their lives* does not have a corresponding *to* form, since a Directional or Goal theme is apparently lacking. Conversely, the *Canada* example or **I threw the ground the ball--I threw the ball to the ground* seems to show that possession is nine-tenths of the Dative Shift law.

The question now is, How can these restrictions be acquired from positive evidence alone? Let us take up each condition in turn, starting with restriction (iii).

There seems to be abundant evidence for an acquisition procedure to associate a thematic role of Goal with a *to*-NP form. Many sentences appear with *to* indicating directionality. If we assume, as before, that the learner can independently recover the thematic role of such NP's, then the right association can be forged:

I walked to the store.

store = Goal (or Direction)

Just how this directionality is recognized remains to be explicated. Perhaps it is recognized as a transfer of location, in some fashion. For example, *I walked to the candy* is bizarre unless *the candy* is construed as denoting a particular point in space. Similarly, *I gave a book to Bill* involves a change in the location of the book, in an extended sense. Speculating, as with the case of phonological acquisition, it could well be that the first step is to recognize that *to the store* does *not* serve in the same thematic role as *a candy* in *I ate a candy*. Only later would this new category be given a "name" of its own, e.g., "Goal." Just how this might happen will be left unsettled here.

In any case, the assignment of a Goal or Directionality theme to the object of a *to* Prepositional Phrase already rules out some problematic cases of Dative Shift. The *to*-NP subcategorization is assigned on a verb-by-verb basis, and occurs in adult speech only if inherent Directionality is thematically assigned. Therefore, *to*-NP forms are never found in conjunction with verbs in which Directionality makes no sense, e.g., *begrudge, cost, envy*.

*I begrudge his wealth to John. (?Transfer of wealth from me to John -- impossible)
*This shirt cost $20 to me. (?Transfer of $20 to me -- impossible.)
This book makes sense to me. (Transfer of sense from the book to me.)

What about the thematic distinction regarding possession? There is reason to believe that this distinction is one of the earliest made by children (Keil [1979]). What things a person (or animate being) can or cannot possess, and what qualities an object can or cannot have are intuitions that children seem to have. This is demonstrated by the fact that the test questions that Keil used in his study were of this form: "Can a tree have a cold?"; "Can a dream be tall?" and so forth. The fact that even very young children have intuitions that these questions are sensible or not means that they can draw a distinction between posssession and non-possession. If this is so, then a child can be assumed to have knowledge designed to distinguish between the following two sentences:

I threw the ball to the ground.
I threw the ball to the boy.

because children seem to be able to answer the questions, "Can the ground have a ball?" "Can a boy have a ball?"

With the "possession" distinction available, one can now attack the problem of how the acquisition procedure can learn when to incorporate an NP, and when incorporation is impossible (given a particular verb).

First of all, incorporation demands adjacency:

I gave the books away to John.
I gave away the books to John.
*I gave away John the books.

I gave up the fugitive to the police.
I gave up to the police the fugitive.
*I gave up the police the fugitive.

What does it mean to incorporate an NP into a Verb? In part, incorporation has something to do with partial, but compositional, semantic intepretation; the verb and its (Indirect) Object NP are formed into a kind of semantic unit, as by Weinberg and Hornstein [1981] in the related case of Preposition stranding. Weinberg and Hornstein pointed out that the existence of prepostion stranding depends in part on the ability of a verb to "absorb" a preposition or prepositional phrase and form a single complex verb:

I decided on a boat.
What did you [[$_V$decide--on] trace]

Who did [$_S$ John [$_{VP}$ [$_V$ talk to Harry about] trace]

This view has a natural interpretation in a model of on-line sentence processing. It amounts to the claim that semantic interpretation (whatever that comes to) is incremental, in the sense that completely constructed arguments to a Verb are interpreted immediately as they are finished, rather than waiting until an entire argument structure (the complement of the V) has been built. This new predicate (not a natural predicate, in most cases) is then applied to the next argument that is completely built, and so forth:

[I [$_{VP}$ [$_V$ gave] [$_{NP}$ a book][$_{NP}$ to John]]]

[I [$_{VP}$ [$_V$ gave a book][$_{NP}$ to John]]]

[I [$_{VP}$ [$_V$ gave a book to John]]]

This process is akin to that of Skolemization in logic, though the analogy should not be taken too literally. As is well known, a function of two arguments, $\Phi(x,y)$, may be replaced with a function of one argument x and an argument f(x) as follows:

1. $\forall x \exists y\ \Phi(x,y)$
2. $\models_{QL} \Phi(x,y)[\text{substitute}(x|a)(y|b)]$ (By line 1.)
3. Set $f(a)=b$ (pairing of a,b determines a function)
4. $\models \exists F \forall x\ \Phi(x, F(x))$ (where *f* is the function that makes this valid)

Incorporation mimicks this procedure. The basic idea -- adopting for a moment a model theoretic view, though this is not crucial here -- is that the complex predicate *give a book* picks out just those possible worlds where *give a book* is true. There is then some function *f* corresponding to this selection. *f* in turn is applied to the remaining argument, *Bill*. Note that Skolemization will also apply to non-Dative shift examples, but in this case a different function would be constructed.

If this interpretation of incorporation is correct, then it is also natural to assume that the compositional formation of a complex predicate -- incorporation -- demands that thematic role assignment be correctly carried out as well. That is, in the adsorption of *a book* into *give*, *book* must be understood as receiving Objective case, so that the thematic role of Patient may be properly assigned; the predicate *give a book* with *book* as Recipient is ill-formed. As we have seen, the proper assignment of thematic role in a verb that has a [NP *to*-NP] subcategorization frame exploits *to* as a Case marker, and an indicator of Direc ionality. Therefore, it appears as though the correct assignment of case and thematic role requires *to* as a marker of some sort. But this marking is absent in the Dativized sentences. If incorporation is to be possible then, there must be some way to recover this deleted case marker; otherwise, the complex In other words, *I gave Bill a book* means *give--to Bill -- a book*, and

this meaning must be somehow recoverable. Thus this condition amounts to a recoverability of deletion restriction on incorporation. This is perhaps the reason for the *inalienable possession* restriction on incorporation. The thematic role of Possessor is sufficient to recover the thematic assignment of Directionality, and so incorporation can proceed. (Note that Directionality need not be recovered unless the verb is known to have a subcategorization frame [NP *to*-NP], however.)[40]

How is knowledge of incorporation acquired? The impossibility of incorporation with verbs such as *report, say, demonstrate* is in fact revealed by positive examples:

*I reported Bill the crime. I reported to Bill that John was sick.
*I suggested Bill the crime. I suggested to Bill that John was sick.
*I said Bill something. I said to Bill that John was sick.

For some reason, the marker *to* cannot be deleted in such cases. According to Stowell, this distinction between e.g. *show--demonstrate* is reflected in the phonological properties of Native vs. Latinate words. Native (Germanic) words are monosyllabic, or, if disyllabic, have primary stress on the first syllable (give, threw, xerox); Latinate words are the complement of this class (persuade, suggest, report). Native words can undergo Dativization; Germanic words cannot. Exceptions seem to involve Latinate words that now receive first syllable stress:

I promised Bill a bicycle. (cf. I promised to Bill that I would not leave).

And neologisms seem to follow this rule, as Stowell observes: I xeroxed him a letter; *I photocopied him a letter.

The retention of *to* seems natural here; perhaps it is a property of Latinate words that they retain a residue of obligatory case marking (in some way that I do not fully understand).[41]

---

40. One could speculate as to how is the Possessor theme might be assigned. It could be derivative of the "typical" Direct Object assignment: I kicked the dock --> I gave the dock a kick. "Possessor" is not quite the right label here. Then any argument string adjacent to the Verb would be assigned this thematic role. Again, there is abundant positive evidence for this choice.

41. Latinate verbs seem to demand a case marker, and take S and NP complements:

I suggested to the police that they look elsewhere.
I reported to the police that a crime had been committed.
I reported for work Tuesday.

(In Stowell's analysis, S is forced to the end of the phrase to avoid double case marking-- so the actual structure is:

[I [suggested][e_i][to the police][that [S]])

There are some apparent exceptions to the phonological classification of Native and Latinate words:

I said Bill a story (one syllable, Germanic) -- but perhaps this is because *say* involves direct "quotation"?

I designed a new bathroom for my mother. I designed my mother a new bathroom. (Latinate, bisyllabic, primary stress on second syllable)

I assigned him the second row. (trisyllabic, stress on second syllable.)
I awarded Bill second place. (Latinate OF)

I will radio Bill the answer. (trisyllabic, Latinate, but stress on first syllable)

What about *donate*? There are several reasons to believe that this word is exceptional. First of all, *donate* is a peculiarly American usage. Second, and more importantly, it appeared historically with the preposition *with*: *The settlers were donated with warrants for land* (OED). Indeed, its Latin root means, "to present X with Y". Just like "give away," then, perhaps donate is really "donate with," in which case the case marker "with" blocks verb incorporation, hence Dative Shift. But how could this subtle distinction be learned? Actually, as might be expected *Bill donated the library the books* is accepted by some speakers (stress has shifted to first syllable).

Finally, for cases such as, *I threw the ground the ball*, observe that it has been assumed that the acquisition procedure can answer "possession" predicates such as, "Can the ground have a ball?" If this is assumed, then note that attempted "Skolemization" of the predicate form *give--Canada--a ball* fails, since the complex predicate *give Canada* is meaningless. (Cf. interpretations in which a denotation can be assigned, e.g., "I gave the United States a hard time.") Thus these examples may be ruled out.

Summarizing, the following sequence of examples and deductions are required to fix what is called Dative Shift:

(0) Establish Direct Objects for the relevant verbs.
Evidence: I gave a book...

(1) Establish the Adjacency requirement for Direct Objects. (English)
Evidence: Default unless proven otherwise.

(2) Establish *to*-NP as part of subcategorization frame of the verb.
Evidence: I gave a book to Bill. (Rules out *cost*, *envy*, and the like.)

(3) Establish directionality of *to*-NP.

Evidence: I went to the store.

(4) Establish the possibility of NP incorporation into the verb.
Evidence: Given the Adjacency constraint, and assuming independent reconstruction of the thematic role of the Direct Object, then *I gave John a book* must collapse *gave -- John* into a V.

Evidence for incorporation itself: I picked up the book.

(5) Establish "inalienable possession" for incorporated NPs, or deducibility of the theme of the indirect object. (Establish possible predications.
Establish possession distinction (possible predicates).
Evidence: Bill has a book. *The ground has a ball.

Now consider the apparent developmental dependencies implicit in this acquisition sequence. The adjacency constraint must come before incorporation is attempted -- but it does, since it is the unmarked, default case. Second, the subcategorization frame of a verb must be acquired before incorporation is attempted, but this too is a constraint that will automatically be met, since without the subcategorization frame one is prohibited from syntactic inferences. The only other dependencies are to acquire the directionality of *to* and the constraints on incorporation. These come after the subcategorization frame is established. If directionality (or thematic role assignment to the Indirect Object) is not understood, then incorporation fails, so this fact must be established before Indirect Object incorporation. If incorporation itself is assumed to be a central part of semantic interpretation, then it must be acquired quite early and may be assumed to be the unmarked case, for Direct Objects. Finally, the notion of inalienable possession seems to be understood at any early age, though it is not clear where it fits into the above sequence.

The Subset Property is invoked in several places: (1) to make the Adjacency restriction the default; (2) to make NP arguments by default obligatory; (3) to demand incremental decision-making (and hence order examples intrinsically).

3.3.1 Other cases of the Subset Property

1. The PRO-drop parameter.
Languages such as Italian exhibit a cluster of properties that languages such as English or French do not: among other things, they permit sentences with missing Subjects, and can invert Subjects freely:

Found the book.
ho trovato il libro ("I found the book")
ha mangiato Giovanni. ("Giovanni ate")

(Chomsky [1981 page 240])

It has been suggested (see Chomsky [1981]) that these differences may be the result of a "parameter" that is set to one value for Italian, and to another for English and French. Since the empty subject is presumably a Pronominal element (see Chomsky [1981] for discussion), this parameter has been called the "PRO drop parameter." Regardless of whether this is the correct analysis of this difference between, e.g., English and Italian, the question arises as to how the setting of the PRO drop parameter be acquired, so that English and French can be distinguished from Italian. For purposes of exposition, let us say that the PRO drop parameter has just two values: 1 (if Subjects can be dropped, and PRO inserted, yielding an Italian type language) and "0" (if Subjects cannot be dropped, and PRO inserted, yielding English or French type languages). One proposal is that a principle of *indirect* negative evidence operates in this case: if no example sentence of a certain complexity or less is found to exhibit evidence of PRO drop, then assume that PRO drop is not permitted at all [Rizzi 1980]. For example, one could adopt the Wexler and Culicover restriction and assert that the entire transformational system is acquired on the basis of sentences of degree of embedding two or less. Then, if no sentence of this depth or less exhibited PRO drop, the child could assume that PRO drop was not permitted. This method plainly works, assuming some kind of complete presentation sequence of all positive sentences less than some fixed complexity. Note that this guarantees that the Subset Property will hold: here, the finite triggering set is the set of positive sentences less than some degree of embedding -- a finite set, assuming now a fixed lexicon.

But there is a simpler approach that also works, and that does not require the assumption of negative evidence at all. Note that a language that permits PRO drop is *broader* than one that does not, since it can generate two surface forms where a non-PRO drop language can generate only one. This fact leads one directly to consider an alternative solution: why not force the Subset Property directly, and insist that the unmarked (default) assumption for the PRO drop parameter be the one that generates the *narrowest* class of languages? Then the default setting would be "0" -- a non-PRO drop language. If the acquisition procedure ever encounters a positive example sentence where the Subject is absent, e.g., *Found the book* (meaning, "I found the book"), then the parameter can be set to "1" so as to generate the larger language. If the acquisition procedure never hears a sentence with PRO dropped, then the parameter remains set where it was to begin with. No negative evidence need be used.

In fact, a more general result obtains. Consider any situation where indirect negative evidence is used to set the values of a parameter P. Without loss of generality, assume that P can have just two values, 0 and 1, and that these parameter values result in grammars that generate different languages, i.e., $L(G|P=1) \neq / L(G|P=0)$. If indirect negative evidence is used to set the value of P, then this occurs in the following way: We assume some default value for the parameter P, say $P=1$. (Actually, as we shall see, if negative evidence is to be used then this value must be set so as to generate a superset language, so we are not entirely free to choose the value of P.) The sentences of the target language are

enumerated according to some complexity measure (say length, or depth of embedding), such that a finite initial segment of the set of all sentences of the target language is enumerated. Given this finite sequence of positive evidence, a sentence is observed *not* to occur, and on the basis of this evidence the acquisition procedure concludes that the value of P should be changed to, say, 0. In other words, since L(G|P=0) does *not* generate certain strings less than some complexity value *c* then when such strings n fact do not appear, then the acquisition procedure concludes that the value of P should be 0. Since this evidence was assumed to distinguish between P=0 and P=1, it must be the case that L(G|P=1) does generate at least one of these non-occurring strings. In short, we have the following conditions are sufficient for indirect negative evidence to apply: (i) a (default) parameter setting that generates some (evidently large) class of sentences or surface forms; (ii) if sentences are *not* generated that should be generated according to the default value of this parameter, an alternative value of the parameter exists that generates a narrower class of languages (based on a finite segment of positive evidence). But observe that this procedure is precisely the dual of acquisition from positive-only evidence according to the Subset Property. According to the Subset Property, hypotheses are ordered so as to start with a parameter setting that generates the narrowest class of languages and then expanded as demanded from a finite positive sample. Indirect negative evidence starts with the largest class of languages, and whittles that class down based on a sample of negative evidence. Given a finite sample, the two approaches are completely symmetrical. Wherever indirect negative evidence can be exploited, one can also simply reverse the order which hypotheses are entertained and use simply positive evidence, according to the Subset Property.

2. S deletion. (Lasnik [1981])
As it stands, the Government-Binding theory would rule out the following sentence as ungrammatical:

John [[$_V$ was expected] [$_{\bar{S}}$ [$_S$ to win]]

Why is this? Basically, it has to do with the interaction between the empty element that is the Subject of the embedded clause *to win* and the constraints that govern where empty subjects can appear. What is the empty category in this case? In the Government-Binding theory, it may be either a trace or PRO. Trace must be (properly) governed, while PRO is ungoverned; this is difference intended to capture the distinctin between the empty category that serves as the Object of a passive sentence and the empty category that can be the Subject of an embedded clause:

Sue was kissed [empty category -- trace]
John tried [[empty category -- PRO)] to win]

In the unmarked case, PRO is obligatory for infinitivals, since a full clause, S, acts as a barrier to government. This fact is revealed by sentences where the Subject of an embedded clause is lexically realized:

John tried [PRO to win]
*John tried [Bill to win]

Since lexically realized elements must be governed, we conclude that the position after *try*, embedded in the S, is ungoverned. Thus the empty category is PRO, not trace.

This cannot be the whole story, however. Sentences such as *I expected [Bill to win]* show that the Subjects of embedded complements of certain verbs, *persuade*, *expected*, and the like may be lexically realized, hence must be governed. If so, then PRO cannot appear, and some explanation must be found for government in these cases. To overcome these difficulties, Chomsky [1980] proposes a "reconstruction" solution similar to that involved in Dativization. Suppose that a marked property of certain verbs is that they permit the bracketing $[_S\ [_S$ to be collapsed to $[_S$. Deletion of the S now permits government of the embedded Subject, hence a trace or lexical NP, as observed.

Again the question of acquisition may be raised, as observed by Lasnik [1980]. Since S-deletion permits a *broader* range of surface strings than no S deletion -- either an empty string or a lexical NP -- it represents a superset hypothesis. Therefore, if the Subset Property is to be maintained, S-deletion should be a marked option; the default assumption should be that S-deletion is not possible. If a verb is encountered where government can be observed, e.g., John expected Bill to win, then the acquisition procedure can assume that S deletion is possible for that verb.[42] Finally, note that an indirect negative evidence solution is possible in the case of S deletion, just as with the PRO drop parameter. The acquisition procedure could assume that all verbs could undergo S deletion; if it then examined all sentences less than two S's deep and observed that no forms such as *John tried Bill to win* appeared, it could conclude that S deletion was impossible for this verb. Exactly as before, however, one can see that indirect negative evidence is not required to set this parameter. This result is expected, given the general theorem that any parameter identifiable from indirect negative evidence can also be set using direct positive evidence.

3. Bounding nodes for Subjacency. (Rizzi [1978])
In most current theories of generative grammar, one assumes that movement rules obey a certain "locality principle", in that a movement cannot cross more than a single clause boundary -- the Subjacency constraint. (This is the case even if explicit "movement" rules are not permitted, but rather interpretive principles that co-index empty categories with their antecedents.)

---

42. Note that the Subset ordering makes a developmental prediction. The acquisition procedure assumes that a verb does not govern across an S boundary unless evidence is obtained to the contrary. If this is correct, and making the usual (difficult) assumptions that acquired rules should also appear in productive use, then (incorrect) forms such as, *I wanted [Pro to win]* should appear before forms that delete S, such as *I wanted Bill to win.* Is this confirmed?

John is certain [$\overline{S}$ trace to like ice cream]
*John seems [$\hat{S}$ it is certain [$\overline{S}$ trace to like ice cream]]

The man [who$_i$ [$_S$ I don't know [who$_j$ [$_S$ trace$_j$ knows trace$_i$]]]]

Interestingly, this last sentence is grammatical in Italian, as discussed by Rizzi [1978]:

L'uomo [*wh*-phrase$_i$ che [non so [chi$_j$ [trace$_j$ conosca trace$_i$]]]]

According to Rizzi, this is because it is $\overline{S}$, not S, that is a bounding node for Subjacency in Italian. Therefore, the empty category *trace*$_i$ can be co-indexed with the *wh*-phrase$_i$ because it crosses only a single full clause boundary; the second boundary is an S, not an $\overline{S}$. Apparently, the choice of a bounding node is yet another parameter that must be set in order to "fix" a grammar.

Suppose Rizzi's analysis is correct. How could the choice of bounding node be determined on the basis of evidence received by an acquisition procedure? Once again, let us apply the Subset Property. If the bounding node for Subjacency is S, then a narrower class of languages is generated than if the bounding node for Subjacency is $\overline{S}$. Therefore, by the Subset Property, the acquisition procedure's first hypothesis should be to set the bounding node for Subjacency to S. In other words, the default assumption is that all languages are like English in this regard. If this assumption is wrong, then a positive example will appear that violates S-bounding -- as in the Italian example above. Then the acquisition procedure can reset the Subjacency parameter to the next "largest" value, namely, $\overline{S}$.

If Subjacency can be weakened from S to $\overline{S}$, then why couldn't there be a language that weakens the Subjacency condition one step further, allowing movement (or interpretation) to cross *two* or more clause boundaries? It may be that this follows from the evident *non counting* property of natural languages. Apparently, the rule systems of natural grammars do not make use of predicates that count; there is no rule that moves an item so that it is three words from the end of a string, or, for that matter, checks to see if it has moved an element over three clause boundaries. It is well known, however, that *string adjacency* is a common predicate in grammars; witness the adjacency condition on Case assignment. Thus natural rule systems can state whether a phrase is adjacent to another phrase or not (and hence, in a derivative sense, can determine if a phrase is "one" clause away), but they cannot do more than this. If *adjacent clause* is a natural predicate and, say, *fifth clause* is not, then one could not even *state* an extra-subjacent condition, let alone acquire it.[43]

If specifying S as a bounding node for Subjacency is a tighter constraint than specifying $\overline{S}$, then it

---

43. See Berwick and Weinberg [1982].

follows that making *all* categories bounding nodes would be an even tighter constraint; if the Subset Property is strictly applied, then this should be the true default setting of the Subjacency parameter. What this would mean is that, in the default case, no movement would be possible out of A, V, P, or N phrases (or all phrases X^{max} of type X). This constraint could be weakened as positive evidence is received that violations can occur, analogous to the case of the S vs. $\bar{S}$ parameter. See Koster [1978] and Baltin [1978] for evidence that a generalized locality principle like Subjacency is operative in domains other than S.

An even tighter constraint than this would be to assume all X^0 (lexical) categories are bounding nodes. In effect, this would mean that no movement would be possible at all; that is, word order would be fixed as determined by any adjacency constraints.

In sum, the Subset Property may be invoked to establish an ordering of hypotheses concerning the surface output of possible movement rules. The ordering is such that the default hypothesis is that there is no movement, since this produces the smallest class of output sentences; each new hypothesis admits an incremental increase in the range of output sentences. The acquisition model outlined in Chapter 2 respects this ordering. Recall that new actions are attempted in the following order:

Attach (without any permutation or movement.)
Switch (the first two items in the buffer)
Drop a trace (into the buffer)

This ordering reflects a hierarchy of decreasing restrictiveness on movement. The first assumption made is that *no* movement will be required, and that the constituent in the first position in the buffer can be attached to the current active node without further modification. If this action cannot succeed because of a subcategorization violation, then a purely local string permutation, *switch*, is attempted.[44] Finally, if the attempted *switch* rule fails, then a less restricted rule, *drop trace* (the parsing equivalent of *Move* α) may be invoked. Note that in the case of both *switch* and *drop trace* that only strictly *adjacent* predicates are invoked: adjacent buffer cells in the case of *switch*, and adjacent S-domains in the case of *drop trace*.

4. Identifying NP empty categories

In the Government-Binding Theory, an NP can have the properties of a name (a *referring expression* or R-expression), possessing intrinsic reference (e.g., *the boy*, *John*); a pronominal, acting as a pronoun (*he*); or an anaphor, acting like a bound variable. These are not mutually exclusive choices. NPs possessing one or another of these properties can be distinguished because anaphors, pronominals, and

---

44. Recall however that the second item in the buffer must be a complete constituent, so that it is not strictly accurate to say that *switch* is a string operation.

R-expressions are subject to different binding constraints. An R-expression, with intrinsic reference, is *free* everywhere, since its "value" is not (and cannot be) determined by binding to another NP. A pronominal such as *she* must be *free* in a certain local domain, namely, the so-called *minimal governing category* (MGC); for our purposes here, we may take the MGC to be simply the S of which the pronominal is a part. This constraint is evidenced by examples such as:

John hit him.

John knows [$_S$ that he will win.

Where *him* is in the same S domain as *John*, as in the first sentence, *he* cannot be bound to *John*; in contrast, in the second example *he* is not in the same S domain as *John*, hence can be bound to *John*.

Finally, anaphors must be *bound* in their MGC -- the dual of pronominals. For example, consider the behavior of *each other*: it apparently must be bound locally:

They like each other. (they = each other)
*They know that [I will like each other] (they = each other)

These possibilities can be ranked in order of decreasing restrictiveness on output surface structures. Thus the *weakest* possible constraints are placed on R-expressions (names), since they may appear anywhere. The strongest constraints are placed on (pure) anaphors, since they must be bound in their local governing categories. But at worst this means that an anaphor is bound within the S (or $\bar{S}$) in which it is located. That is, an anaphor that has a governing category at all is bound within a finitely specifiable domain;[45] it cannot appear bound across arbitrarily large domains such as the following:

*[They$_i$ knew that I felt that John wanted.... to like each other$_i$]

But this is precisely the set of domains where pronominals *can* appear:

[They knew that I felt that John wanted .... to like them]

One can see then that the set of surface structures (now interpreted in an extended sense to include co-indexing) where pronominals can appear is larger than the set of surface structures in which pure anaphors appear. One could, therefore, invoke the Subset Property and establish an order in in which hypotheses about the properties of an unknown NP element will be made. The tightest assumption is that an NP is *bound* in its MGC; hence, the default assumption should be that an NP is an anaphor (unless explicit positive evidence exists to the contrary). So for example, suppose that *each other* is

45. Some anaphors, such as PRO, do not have a governing category, and hence are not so restricted; see below.

encountered for the first time. By default, *each other* will be assumed to be an anaphor (as is correct in this case).

They liked each other.

Since no example where *each other* is not locally bound will ever occur, this first default hypothesis will be retained. Note that if this is correct, then literally *nothing* must be "learned" about *each other*, the optimal situation.

This analysis is not forced, however, since in fact pronominals and pure anaphors are in nearly complementary distribution. (Just where pure anaphors appear, pronominals cannot appear, and vice versa.) This means that there are simple positive examples that distinguish pronominals from anaphors, so one should be able to start with either assumption as an initial guess, and then use a principle of indirect negative evidence to resolve over-general guesses. Suppose for example that one opts for the dual strategy of the one just suggested, that is, assume that all elements are names. Then by default it would be assumed that *me* and *each other* are names, and hence are free everywhere. Then the following surface forms would be generated.

John likes Bill.
John likes me.
I like Bill.
I like me.
Bill likes them.
John and Bill like each other.
etc.

There is some reason to believe that this is in fact the strategy that children adopt; for example, Baker [1979] observes that children use *it* as if it were a full NP in constructions such as,

I turned off the light. I turned the light off. I turned it off. I turned off it.

Of course, this overly-general guess is wrong. In this case, however, there is indirect negative evidence that can serve as a corrective. If the element is a pronominal like *me*, then it will never appear locally bound. The non-appearance of forms such as *I like me*, together with the assumption that *me* is known as an element that references *I*, is sufficient to inform an acquisition procedure that local co-indexing of pronominals is impossible. Note that this is an inference based on indirect negative evidence, since the conclusion is drawn by observing that certain sentences do *not* appear. As expected from the general result established above, this is the dual case of the hypothesis sequence that uses positive-only evidence.

What about the null phonetic pronominal, PRO? Recall that there are two possible "realizations" of

empty (phonetically null) categories, as trace or PRO. Trace and PRO differ in that trace is governed, and PRO ungoverned:

John was kissed [trace]
They thought I said that [PRO feeding each other] would be difficult.

One might think of trace or PRO as simply a cluster of properties: trace=[-phonetic, +governed]; PRO= [-phonetic, -governed]. Let us assume for the sake of exposition that the acquisition procedure has determined that S (in general, X^{max}) serves as a an opaque boundary for government. How could an acquisition procedure distinguish between trace and PRO? Once again, first suppose that the default assumption is the one that generates the narrowest class of languages. What assumption about government is the more liberal? Suppose every empty category was ungoverned. Then output forms such as the following could not appear:

John was kissed [empty category] --> empty category is trace.

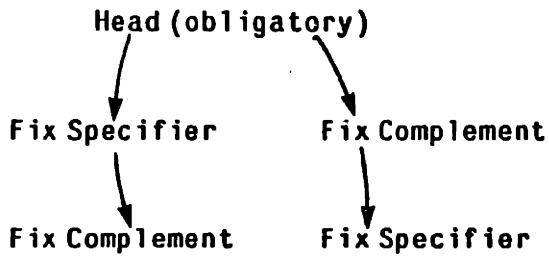
But forms such as this could also appear:

They thought I said that [empty category] feeding each other would be difficult.

If PRO is ungoverned, it vacuously meets the condition of being free in its MGC, since it has no minimal governing category. But then, there is an infinite class of surface structures that exhibit PRO as co-indexed to some other element. Conversely, if empty categories are assumed by default to be governed, then they can appear is a more restricted set of environments. Superficially at least, it would seem that the default assumption should be to assume empty categories to be governed until proven otherwise.

5. X-bar acquisition of Specifiers and Complements

Williams [1981c] describes a special case of the Subset Property as a potential model for the acquisition of X-bar Specifiers and Complements. This model was also independently developed in the design of Lparsifal. The idea is simple. Recall that according to the X-bar theory, a phrase structure rule obligatorily contains a Head, a unit of the same type as the phrase itself, and optionally has Complements and Specifiers. For example, in an English Verb Phrase the Verb itself is the ultimate (lexical) Head; the trailing NPs and PPs, and S are the Complements; perhaps Adverbs are Specifiers. Given this model, then the Subset Property dictates a particular order in which hypotheses should be entertained. The first assumption is to assume that a phrase has no Specifiers or Complements, since this generates the narrowest possible class of languages. As usual, this initial assumption would be dropped in the face of positive evidence to the contrary -- in this case, the existence of a Complement or a Specifier. Thus the order of hypotheses would be as follows:



As Williams observes, there is no particular order in which specifiers or complements are fixed, relative to each other. Further, since either or both can be missing, the end state may be a phrase structure rule lacking in either Specifiers or Complements. More often, however, Complements are obligatory arguments; hence the general order of acquisition will be to fix Complement structure before fixing optional Specifiers.[46]

In Lparsifal, this model was originally implemented by means of three "meta variables" X-SPEC, X-HEAD, and X-COMP, associated with every category of type X. (See Chapter 2 for additional discussion.) The values of these variables are initially set to nil, and the data structure consisting of {X-SPEC, X-HEAD, and X-COMP} is an unordered set. As described in Chapter 2, positive example sentences force this data structure to become an ordered one; e.g., in English, (X-SPEC X-HEAD X-COMP). This order is set cross-categorically, in accordance with X-bar theory. As a result one can use evidence from disparate examples to set different parts of the ordering; for instance, the NP fragment *a green ball* could be used to set the relative order of X-SPEC and X-HEAD, and a VP fragment *kiss the boy* sets the relative order of X-HEAD and X-COMP. Once this order is established, the acquisition procedure uses the resulting data structure as a template that activates and deactivates packets of grammar rules (as in the original Parsifal design). Entry in the phrasal data structure is prompted by the detection of *any* valid value for one of the three meta-variables -- either a Specifier, Head, or Complement. For instance, the Specifier *the* triggers entry into an N phrase (as in Marcus' design); this amounts to the top-down *prediction* that a phrase of type N will be constructed.[47] Generalizing this approach, detection of a V, as in *kiss the boy*, likewise triggers entry into a V phrase. In Parsifal, this top-down prediction was handled by creating a VP node and immediately a V node, attaching the V to the VP, via a single grammar rule. In Lparsifal this process has been uniformly factored out, so that VP creation occurs just as it does with NPs: entry into the X-phrase data structure signals that a node of type X will be built, so one is created and placed on the active node stack.

---

46. In this regard it is interesting to point out that children seem to frequently drop Subjects, behaving as if the language was a PRO drop language. It has been suggested that this is an output (production) constraint of some sort (see Newport, Gleitman, and Gleitman [1977]). Apparently, obligatory arguments -- Complements -- are not dropped as frequently.
47. See Chapter 5 for a more formal analysis of the use of top-down prediction in the Marcus parser.

Similarly, completion of a phrase is signalled by "running off" the end of the X-COMP (that is, by executing all grammar rules in the packet associated with X-COMP).[48]

6. Optional and Obligatory Arguments and Rules

In the discussion of the acquisition of "Dativization" it was pointed out that the Subset Property implies that the arguments to a Verb (more generally, any Specifier or Complement) should be considered obligatory until positive evidence is received that indicates otherwise. This proposal has also been advanced by Roeper [1981]. More generally, it has been widely pointed out (see e.g., Baker [1979]) that it is difficult, if not impossible, to determine whether a rule is obligatory or optional on the basis of just positive evidence. For instance, Baker observes that Subject-Auxiliary inversion is obligatory in main clauses. If an acquisition procedure guessed a Subset language and assumed that inversion was optional, it could generate two forms, the second ungrammatical:

Who will John kiss?
Who John will kiss?

In part this difficulty can be side-stepped because Lparsifal *analyzes* rather than *generates* sentences. Since Lparsifal will only encounter inverted main-clause questions (for whatever reason), its knowledge of the language (as embodied by rules of analysis) in effect presumes that the inversion rule is obligatory, as desired. Suppose counterfactually that the inversion rule was optional in main clauses. This causes no problem; the acquisition procedure will simply acquire a new rule of analysis to handle that situation as well. In short, one can see that by formulating the acquisition procedure as one of fixing new rules of analysis, one automatically orders hypotheses so that the default assumption is that

---

48. A crucial part of this procedure is the ability to locate the Head of a phrase. This means distiguishing the *boundaries* between Specifiers and Heads, and between Head and their Complements. See Chapter 2 and the next section for details.

Moreover, the phrase completion procedure runs into a well-known problem: the Complement of one phrase can be part of another, higher phrase, as in the notorious PP sequences. See Frazier [1979]; Ford, Bresnan, and Kaplan [1981].

The tourists shouted to the guide that they couldn't hear.
Ford, Bresnan, Kaplan [1981]

In this case, the sentential complement can modify either *guide* or can be part of the complement of *shout*. While such cases pose interesting problems from the standpoint of parsing, they do not seem to be a particularly acute problem for *acquisition*. For there will be other positive example where the complement of *shout* and *guide* is unambiguous:

The tourists shouted that they couldn't hear.
I wanted to see the guide that they couldn't hear.

Thus one can acquire separately rules that, acting in concert, yield difficult parsing problems. As discussed in Chapter 2, there is nothing at all mysterious about this. For example, the rule NP-->N S is easily acquired from an example such as, "John saw the cat that the dog bit". But this same NP rule in Subject position leads to well-known processing difficulties -- "The cat that the dog bit died."

Examples such as these simply reinforce the conclusion that it cannot simply be considerations of efficient *parsing* that determine what systems of linguistic knowledge can or cannot be acquired.

a rule is obligatory until proven otherwise. This ordering satisfies the Subset Property.

From a different point of view, Baker's criticism seems unfounded. Suppose that indirect negative evidence is permitted. In the example above, if a non-inverted main-clause question is not heard, then an acquisition procedure can assume that non-inverted questions are ungrammatical. (The dual approach, using just positive evidence and the Subset Property, was the one described in the previous paragraph.) In fact, all of the "projection puzzles" listed at the end of Baker [1979] are solvable in exactly this way, either by the use of positive evidence and the ordering imposed by the Subset Property, or by the use of indirect negative evidence.[49]

7. Boolean Conditions on Rules

Lasnik and Kupin [1977] also proposed to eliminate arbitrary Boolean conditions on rule application (continuing a line of discussion suggested by Bresnan [1976]). Disjunctive triggering conditions of the following form were to be ruled out:

IF [A or B or... C]
THEN <action>

This is clearly in keeping with the "avoid disjunction" principle described earlier. A disjunctive trigger is an unnatural predicate.
a conjunctive trigger,

IF [A and B and C]
THEN <action>

Furthermore, one must also exclude the possibility of complementation:

IF NOT [A and B and C]
THEN <action>

The reason, of course, is that such a predicate would be logically equivalent to:

---

49. A list of these puzzles:
(1) Negative exceptions to transformational rules (E.g., *I reported John the accident). This is solved via the trigger, *I reported to Bill that-S.*
(2) Positive absolute exceptions to transformation rules (E.g., *John tried for Bill to win) This is solved via the failure of S deletion in this case, as ordered by the Subset Property.
(3) Obligatory status for some transformational rules, optional status for others. (E.g., making Subject-auxiliary inversion obligatory in main clauses). This puzzle was just analyzed above.
(4) Extrinsic rule ordering.
(5) Language specific filters.

IF [¬A or ¬B or ¬C]

-- a disjunction once again.

Finally, since existential quantification amounts to infinite disjunction, it too is excluded.

8. The Uniqueness Principle (Wexler)

Chapter 2 briefly discussed the following acquisition principle proposed by Wexler (see Lasnik [1981]):

> Unless proven otherwise, assume that there is just one surface structure for every deep structure.

Note that this constraint has the effect of enforcing the Subset Property, since whenever possible just one surface string is permitted instead of two.

As discussed in Chapter 2, the Uniqueness Principle can be used to force the acquisition of surface filters, such as the *for-to* majority dialect filter of English:

> *I want for to go. I want to go.

3.4 X-bar theory and Phrase Structure Acquisition

The previous sections of this chapter outlined a theory of program size complexity and a general developmental theory of acquisition based on positive only evidence. This section turns to the examination of a specific model of acquisition for the base phrase structure rules of a grammar. This model will combine the constraints of X̄ theory along with a simple extension of finite state automata induction suggested by Joshi and Levy [1977a]. The complexity of this model will be analyzed; as a case study, the English auxiliary system be used, and the results compared to those actually achieved by Lparsifal. Finally, a tentative model for the acquisition of the base of a presumably non-configurational language (Japanese) will be advanced, based on the notions of skeletal parsing using a precedence matrix. Here, the core notion of *government* will be shown to play a crucial role, determining the basic skeletal tree structure of phrases.

3.4.1 Overview: the role of non-terminals in phrase structure

What does it mean for an acquisition procedure to acquire the "base rules" of a grammar for a language? In the Government-Binding theory one way to view so-called "D-structure" is as a representation of pure "GF-θ" -- a canonical association of grammatical functions (such as Subject and Object) with thematic roles (such as Agent, Patient).

John ate the cake

GF: Subj Obj

thematic roles: Agent Patient

Thus at D-structure there is an alignment between syntactic structure and thematic structure, between grammatical functions and thematic roles.

In order for an acquisition procedure to identify *canonical* predicate argument structure, it must be able to determine when an argument is *not* in canonical θ position. In languages where an adjacency requirement on arguments holds -- such as English -- then displacement is in general locally string detectable:[50]

John was hit [___]

I tried [___] to win

In non-configurational languages, displacement at the level of the surface string cannot be the means by which non-canonical argument structure is encoded. Instead, as observed by Chomsky [1981], languages such as Japanese seem appear to add a verbal suffix that triggers a different assignment of GF-θ. This morphological ending is also locally string detectable; for additional discussion, see below. For now, however, we shall consider just the case of languages where some kind of adjacency requirement on Government holds, such as English.

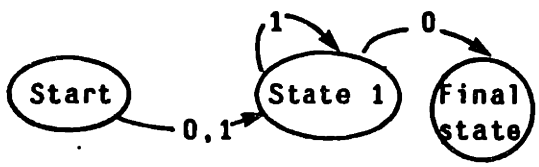
What information is contained in a base (context-free) rule system? There are two basic elements to any such set of rules: (1) the non-terminal labels themselves (e.g., NP, VP, INFL,...); and (2) the constituent boundaries of phrases (in surface strings). Together this information is sufficient to recover the derivation trees of all surface strings, and hence the underlying grammar used to derive

---

50. There are ambiguous cases, as Fodor [1978] observes. But these will then simply not contribute to acquisition. The individual components that make up these cases can be acquired separately.

those strings.[51]

While the "function" of the bracketing or phrase structure of a base rule system seems clear, the reason for the existence of non-terminal labels is not so evident. Phrase structure labels such as "NP" or "VP" are generally assumed to have a transparent semantic interpretation, or are supposed to enter into syntactic operations (such as movement) in a unified way. In other words, the notion of a phrase label is just an informal shorthand for a certain collection of strings that seem to *behave alike*, where the notion of behavioral similarity can range from some kind of vague, semantic grounding (NPs as "objects") to a specific syntactic relation (identity under movement). In any case, the notion of a phrase is derivative, in the sense that it is definable with respect to other grammatical terms. More generally, there is precisely <u>one</u> way in which a "phrase" is formally defined (though this definition appears in many different guises in the literature), and that is as the *equivalence class of a set of elements under some (to be defined) invariance relation.* For instance, if the invariance relation is defined as right-concatenation, so that xRy iff for all z in Σ^*, xz and yz are both in the same equivalence class, and if further there are a finite number of such classes, then it is well known (by the Myhill-Nerode theorem that the resulting equivalence classes are just the states of some finite state machine M. Here, a "phrase" is just a sequence of tokens that puts M through the same sequence of states. For example, consider the finite state transition diagram below. The strings "01" and "111" both cause M to arrive at the same state, namely State 1:



Both 01 and 111 constitute "phrases" of the same type, namely, the phrase denoted by State 1. Behaviorally -- that is, looking at a system of equivalence classes from the "outside" -- the only way to tell if two strings are in the same class or not is to see if they behave alike or differently. Given just surface strings (with no bracketing), there is only one way to do that: strings s_1 and s_2 are *different* just in case there is some string w (perhaps null) such that s_1 w is <u>*not*</u> is ungrammatical and $s_2 w$ <u>*is*</u>

---

51. To see this, suppose that all surface strings can be correctly bracketed and labelled by converting every rule of the form, A-->α to A-->$[_A\ \alpha]$ e.g., "the boy hit the girl that I know"--> $[_S\ [_{NP}[_{Det}$ the$]\ [_N$ boy$]]\ [_{VP}\ [_V$ hit $[_{NP}\ [_{Det}$ the$]\ [_N$ girl$]]\ [_S\ [_{Comp}]$ that $[_{NP}\ [_N$ I$]\ [_{VP}\ [_V$ know$]]]]]]$. This is the information assumed under points (1) and (2) above. But now the derivation trees of sentences (hence rules of the grammar) can be recovered: one can simply scan the resulting bracketed string, constructing a directed acyclic graph with vertices labelled by the subscripts on the left brackets, terminals consisting of the non-bracket items. Start with stack containing "Root"; if input is a left bracket or a lexical item, link top of stack element to the input; if a left bracket, in addition push the input item onto a stack. If the input is a right bracket, pop the stack.

grammatical, or vice-versa. Since accepting (grammatical) and non-accepting (ungrammatical) states must be distinct, a behavioral difference has been forged. This is the basic means by which the equivalence classes of a system can be determined. Note that *any* kind of relation could be used to establish the equivalence classes, as long as that relation separates strings into accepting and non-accepting classes. In the Wexler-Culicover model for the acquisition of a transformational grammar, the relation is in effect the set of transformational rules; a basic claim of the theory they develop is that all strings separable via the adult's set of transformations are separable on a *finite* domain, that is, on trees of limited depth. (More precisely, trees with at most two embeddings of "S" nodes; see Chapter 5 for additional discussion). As discussed below, this is the transformational analogue of the better-known result that the strings of a regular language with an associated minimal finite-state machine with n states are separable by examining initial segments of those strings of length less than $2n$. This characterization of the Wexler-Culicover work will play an important role in the connection between parsability and learnability covered in Chapter 5.

As is also well known, the states of a finite state automaton correspond in a direct way to the non-terminal symbols of a (strongly) equivalent grammar, namely a grammar with rules of the form, A-->a B, where a= a terminal element that prompts a transition from state A to state B. There is a sense in which this associated grammar strongly mimics (simulates) the machine after which it is designed. Using standard terminology, one can say that a grammar is weakly equivalent to some machine M if L(M)=L(G); that is, the grammar G generates exactly the same set of strings that M accepts. This is the weakest possible restriction on the association between G and M; one might call this a state-opaque relation between G and M, because the internal details of the machine are not included. At the other end of the scale, one could insist on a state-transparent relation -- that there be a one-to-one mapping between the sequence of operations of M in the acceptance of any sentence in L and sentential forms in the generation of that sentence in G; this is what occurs in the standard "simulations" of a fsa via a right- or left-linear grammar. In between these two extremes lie a variety of more to less opaque restrictions on the G-M association. For example, suppose that a sentence in L(G) is derived via a sequence of productions, $\{p\} = p_0, p_1, \ldots, p_k$, and accepted by a sequence of machine steps, $\{m\} = m_0, \ldots, m_l$.[52] One could then insist that M *preserve* derivation sequences for all sentences in L(G), by demanding that there exist a fixed non-erasing homomorphism from {p} to {m}. If such a homomorphism exists, let us call the G-M association natural, otherwise, unnatural. We have already exhibited a "natural" G-M association, namely, the standard one that obtains between finite state automata and right- or left- linear grammars. As another example, consider the

52. There are many alternatives here. If the ordering of {p} is preserved, then there are two basic possibilities: one can either retain the sequence of {p}, or reverse it. For example, suppose that {p} denotes a canonical derivation, say a right-most derivations. Then a mapping from {p} to {m} that preserves right-most derivation chains is commonly called a *right cover*. If the mapping is from right-derivations to left derivations, then one has a left-cover.

usual simulation of a context-free grammar G in Greibach normal form via a push-down automaton. Recall that a grammar in Greibach normal form has all its rules of the form, $A \rightarrow x\beta$, where x= some string of terminals, and β= some string of nonterminals, possibly null. In the standard construction, one writes a one-state PDA that simulates left-most derivations in G via rules of the form, $\delta(q_1, a, A) \rightarrow (q_1, \beta)$ whenever $A \rightarrow a\beta$ is a production of G. Clearly then, M will at least *preserve* the derivation steps of sentences in G.

As an example of an unnatural G-M association, consider the following example from Aho and Ullman [1972 page 272]. The following grammar generates the language $aa^+b \cup aa^+c$:

1. S-->BAb 2. S-->CAc
3. A-->BA 4. A-->a
5. B-->a 6. C-->a

Aho and Ullman show that there is no natural deterministic parser for this grammar that outputs a left-most derivation. However, as they point out, one can parse this language deterministically and still produce a left-most derivation: simply store the *a*'s on a stack until either a *b* or a *c* is seen; then and only then output the sequence of productions that were used to generate the string, either 1 5 (3 5)^n 4, or 2 6 (3 5)^n 4, popping the *a*'s from the stack in order to count to *n*. But now note that there is no natural correspondence between the derivation sequence that is output and machine steps, because there can be no fixed homomorphism between {p} and the *n* times that *a* is stored:

```
Derivation sequence:              1     5   3   5   3   5   3   5  4

Machine  |<--- store------>|  b  |<------  Pop  a's ------->| output|
steps:            a's      read
          (n steps)     (output 1)    (each a= emit 3,5)      4
```

Summarizing, the non-terminals of a grammar just denote the states of some machine. Machine states, in turn, are just a way of distinguishing among equivalence classes of externally provided strings.

3.4.2 Finite state induction -- a Review

The relationship between the non-terminals of a grammar and the equivalence classes of some machine accepting or rejecting strings generated by that grammar suggests that the problem of grammatical inference may be reduced to that of inducing the equivalence classes of some machine, given the machine's input-output behavior. This problem is a familiar one in automata theory. First of all, it is well known that one must at least supply the acquisition procedure with an upper bound on

the number of states in the resulting minimal finite state machine. [Moore 1956]. This is because without such a bound an acquisition procedure could not limit in advance the length of strings it must look at. For example, assuming an exhaustive presentation of all positive examples, one could not distinguish between the language a^* and a^n unless the string a^{n+1} appears, or fails to appear. Unless a bound on the number of states is provided, an acquisition procedure will never know when to stop looking. Once the number of states is given, however, one can obtain a finite acquisition procedure by invoking a theorem stating that a finite state automaton is completely characterized by the strings it accepts of length 2n-2 or less.[53]

Suppose then that we are given *n*, the number of states in the minimal (canonical) finite state acceptor

---

53. To prove this familiar result, first recall that two strings s_1 and s_2 are equivalent if they both map to the accepting (or rejecting) state. Let *n* be the number of states in the minimal (reduced) finite state machine M for the language under consideration. Given a string s_1, then there is some string s_2 of length $\leq n-1$ that is equivalent to s_1. For if M has only *n* states, then in accepting a string longer than length *n* some sequence of states must be repeated, i.e., $\text{state}_i = \text{state}_{i+k}$, for $i+k>n$. But then we can "splice out" this sequence to obtain a string that is still accepted (or rejected), hence still equivalent to s_1. Continuing in this manner, one obtains an equivalent string that causes the automaton to pass through no duplicated states. But then, this string can be of length at most *n-1*, since a non-duplicated list of states *n* long corresponds to a string of length *n-1*.

Next, observe that s_1 is equivalent to s_2 iff for all strings *s* of length less than *n-1* the state reached after processing $s_1 s =$ the state reached after processing $s_2 s$. ($\delta^*(s_1, s) = \delta^*(s_1, s)$), where s_1, s_2 denote the *states* reached after processing s_1, s_2.) There are two cases to consider. (i) the states (equivalence classes) reached by the finite state automaton after processing s_1 and s_2 are the same. Then there is nothing to prove, since once the prefixes of two strings enter the same equivalence class, their behavior must be the same on an identical suffix. (I.e., $\delta^*(q_1, s) = \delta^*(q_2, s)$, if $q_1 = q_2$.) (ii) $q_1 \neq q_2$, i.e., the states of the finite automaton after processing s_1 and s_2 are distinct. Then, given that 2 equivalence classes have already been distinguished, at most n-2 more equivalence classes can be distinguished (otherwise, the automaton would have more than *n* equivalence classes, a contradiction). But then, by the same reasoning as in the previous paragraph, M must repeat some state in its processing of the suffix *s*, but this time only n-2 new states are allowed, since q_1 and q_2 were assumed distinct. These duplicated states may be spliced out until a string is obtained of length n-2 that either separates the two strings or confirms them as equivalent.

Combining these two results, any string is equivalent to some string of length *n-1* or less, and adding suffixes of length at most n-2 suffices to distinguish any two strings given that we have already determined whether the prefixes of the strings cause M to wind up in the same of different equivalence classes. Therefore, we need only to look at strings of length *n* to obtain the equivalence classes of strings s_1 and s_2 of length $\leq$n-1, and strings of length n-2 to separate these strings:

Start|----- n ----|---- n-2 ----|
(to obtain equiv. classes of strings) (to distinguish strings)

An examination of all strings of length 2n-2 or less suffices to induce the automaton.

for some regular target language, along with a sample consisting of all strings of length $\leq 2n$.[54] Note also that this induction assumes a principle of indirect negative evidence. That is, the inference procedure must know which strings in the sample are in the target language, and which are not. If only positive evidence is given -- all sentences in the language less than length 2n -- then an acquisition procedure can assume that the strings it has *not* seen are not in the target language, and thus obtain a complete negative and positive sample of sentences less than length 2n. It can then proceed in one of two ways, either by assuming that all strings are in separate states to begin with and then *merging* equivalent states, or by assuming that all strings are in one of two states (namely, the accept and reject states) and *splitting* these states into new ones:

(1) Assume that all positive strings are in different equivalence classes, then find strings that should be merged into the same class. This this essentially the method proposed by Biermann and Feldman [1972]. As demonstrated above in footnote xx, two strings are in the same equivalence class if they exhibit the same behavior on common suffixes of length n-1 or less. For a complete positive sample of strings of length less than 2n, (denoted, $S^+/2n$), identical behavior implies that $\forall\ s_1, s_2 \in S^+$, $s_1 \equiv s_2$ iff $\forall s$, $|s| < n-1$, $s_1 s = s_2 s$. In Biermann and Feldman's terminology, the *n-1 tails* of s_1 and n_2 are the same. (In general, Biermann and Feldman define the *k-tail* of a string s_1 with respect to a positive sample S^+ as $\{s|\ s_1 s \in S^+, |s| \leq k\}$. Denote this set by $T_k(s_1)$.) Thus to infer the states of the automaton, one need only collapse into the same equivalence class those strings of S^+ that have the same n-1 tails. Denote by Q this set of equivalence classes (set of strings). To infer the transitions, observe that a transition carries the automaton from a given equivalence class or possibly to the single rejecting state. Thus for $q \in Q$, $\delta(q, a) = \{q' \in Q | q' = T_{n-1}(sa), s \in Q)\}$, and $q' =$ reject state if $sa \notin S^+$, length $\leq 2n$, for $s \in Q$. The resulting automaton may be non-deterministic and non-minimal; to

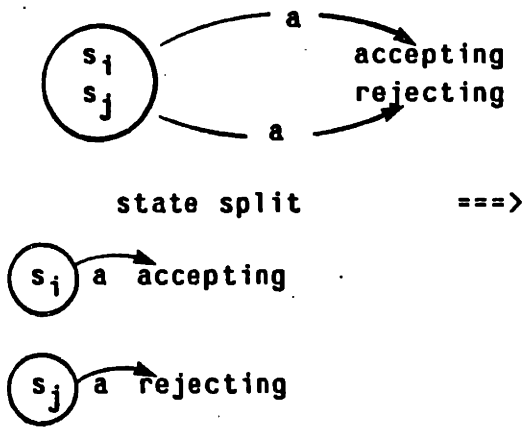
---

54. The assumption that *n* be provided in advance is a significant one. It is not obvious why this information should be available to the language learner. In effect, one is assuming that the number of "natural classes" is given in advance. For the induction necessary to fix an X-bar system, this may not be such a bad assumption, because the basic categories might be simply the lexical possibilities set by the features +/-N, +/-V (determining the core X^0 elements), and the "meta-variables" Specifier and Complement. If these possibilities constitute the full range of non-terminal elements, then there at most 2 x 2 + 2= 6 possible non-terminals. The number of non-terminals is much greater in a theory that assumes a single representational level for phrase structure, e.g., the Gazdar [1980] augmented phrase structure system. Since Gazdar in effect encodes the possible outcomes of the rule "move alpha" into a single level of phrase structure, there are a correspondingly larger number of possible complex non-terminal categories: one denoting the case where an NP is moved to COMP position, as in "Who did Bill kiss"; one for fronted PPs, etc. By collapsing together the levels of phrase structure and the effects of movement in this way, the number of possible non-terminals actually increases multiplicatively: instead of having to fix, say, 4 phrase structure rules and 4 transformational rules, each independently, 4 x 4 rules must be fixed. Of course, one can always *replicate* the additive modularity advantage of a multiple level linguistic representation by stating in an augmented phrase sdtructure system that there is a basic level of rules stated in the non-complex terminal format, and a separate mechanical procedure for producing all legitmate output rules the indicate the locations to which items can be moved, but this seems to be just a way of re-expressing the rule move-α. See Chapter 4 for additional discussion. There are other problems with using automaton induction procedures as valid models of acquisition; these objections are discussed below.

form the minimal (canonical) deterministic machine, one simply applies the usual subset construction and a standard minimization algorithm. [e.g., Aho and Ullman, 1979].

(2) Assume that there are just two states to begin with, the accepting and the non-accepting states; these states are split to yield the required number of states. Thus initially all positive strings go into the same equivalence class -- namely, the accepting state; all negative strings are associated with the non-accepting equivalence class.[55] Given this initial partition, a new partition is constructed by looking for a pair of strings, s_1 and s_2, such that s_1 and s_2 are in the same initial partition (i.e., they both lead to accepting states), but concatenating a single new symbol onto s_1 and s_2 puts them in classes already known to be distinct. This approach follows the method of proof used in footnote xx to show that the equivalence relation $s_1 R s_2$ can be settled by looking at strings $s_1 s$, $s_2 s$ for strings s of length n-2 or less.

More precisely, the only classes initially known to be distinct are the accepting and non-accepting states. Call these q_a and q_{na}, respectively. All strings in $S^+ / \leq 2n$ are initially in q_a and all strings not in S^+ are in q_{na}. Now the acquisition procedure runs through all strings in S^+ to find a pair of strings s_i, s_j and a single new token a such that $s_i a$ and $s_j a$ wind up in different equivalence classes (initially, the accepting and rejecting states). If no such pair can be found, then the acquisition procedure stops, because the equivalence classes need be split no further. Suppose then that such a pair of strings and a token a is found. Then a new state is formed that distinguishes s_i and s_j:

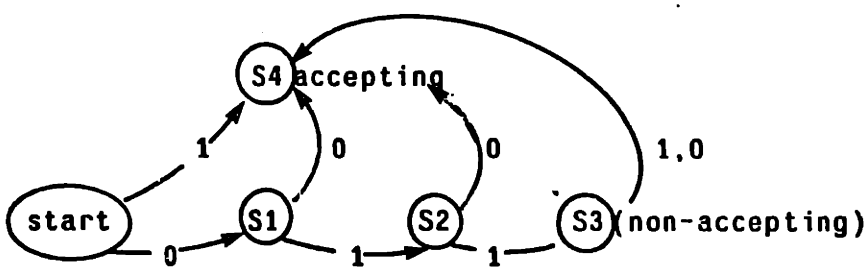


The new partition of equivalence classes includes the original accepting and non-accepting states; s_i; and s_j. The acquisition procedure is now attempts this new set of classes by repeating the above procedure; it again attempts to find a pair of strings $s_{i'}$, $s_{j'}$ such that a single new token a concatenated

55. As before, this assumes a single non-accepting state -- as is the case for a reduced (minimal state) automaton.

to s_i, and s_j, yields distinguishable states -- either s_i, s_j, or the accepting or non-accepting states. Informally, one can see that this procedure eventually terminates, since after each new partition we can consider strings one token longer than the last, and there are but a finite number of such strings that can be considered; moreover, there can be at most n-2 new partitions.

Are there any differences between these two approaches? One way to compare them is to analyze their complexity as a function of the number of examples that are required to arrive at the correct automaton representation for a given (regular) target language. At first glance, it would seem that the two approaches are on an equal footing. Since the number of strings of length less than 2n is of order 2^n, both methods require that an acquisition procedure examine an exponential number of examples, in the worst case. The class-merging procedure, for example, must examine *all* positive example sentences less than a certain length, in order to form the k-tail equivalence classes. For certain families of languages this can amount to an examination of an exponential number of strings. For instance, consider the case of a family of languages $\mathcal{L} = \{L_n | \text{ all strings over } \Sigma \in L_i \text{ except for a single string } s^* \text{ of length n}\}$. The reduced finite state automaton for L_i has n+2 states. An example is shown below, with n=3, and the string 011 as the the only non-accepting string. Note that all strings have the same k-tails *except* for the strings 0, 01, and 011.[56]



The acquisition-by-string merger procedure must in effect find this single non-accepting string by locating these three strings that fail to have the same k-tails as all other strings. But then, given only a list of positive examples less than a certain length and the state bound $n+1$, in the worst case the procedure would have to examine an exponential number of positive examples before finding that the k-tails of λ (the empty string), 0, 10, and 011 are different from the k-tails of all other strings.[57]

Similarly, the refinement procedure must look at an exponential number of examples in certain cases. Consider the dual of the languages considered above -- that is, where precisely one string of length *n* is

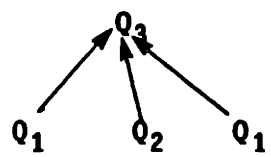
---

56. This is the dual of a case suggested by Angluin [1980].

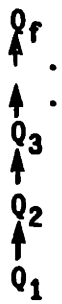
57. Of course, the relevant information could be placed (by chance) at the head of the list of positive examples -- as in this example -- in which case three different states would be discovered immediately. However, one can easily design a language that places the tell-tale examples at the end of a list ordered by length: simply make the languages L_i consist of all strings less than length *2n*, but for one string of length *2n* that *ends* in a particular string *l* of length *n*.

in each language L_n. Then unless essentially all strings of length n are examined, the acquisition procedure must sometimes mis-identify a string. For suppose that in all cases the acquisition procedure uses fewer than 2^n examples. Then there must be at least two strings that are never examined, call them n_1 and n_2. But then the acquisition procedure cannot distinguish between L_{n_1} and L_{n_2}.

So far the analysis has focussed only on the induction of finite-state automata. However, it is easy to show (as suggested by Joshi and Levy [1977a] and Brayer and Fu [1977]) that the same partition or merger algorithms can be applied to the induction of context-free grammars if one uses tree automata instead of finite-state automata. Recall that the transition function of a finite state automaton is defined as a mapping from states and input symbols to a new state -- $Q \times \Sigma \rightarrow Q$. One can therefore imagine the succession of states that a given finite state automaton enters to form a straight, non-branching chain, $q_0, q_1,...,q_f$. A tree automaton is a simple generalization of this kind of next state mapping, from sets of states and input symbols to new states: $Q^* \times \Sigma \rightarrow Q$. Graphically, one may imagine one transition of such a (frontier-to-root) tree automaton as moving from the daughter nodes of some node a directed acyclic graph to the common mother node. For example, if the transition is based on three states, $Q_1 \times Q_2 \times Q_1 \rightarrow Q_3$, then the picture would look as follows:



Observe that a finite state automaton is indeed a degenerate case of a tree automaton, where the "tree" consists of a straight line:

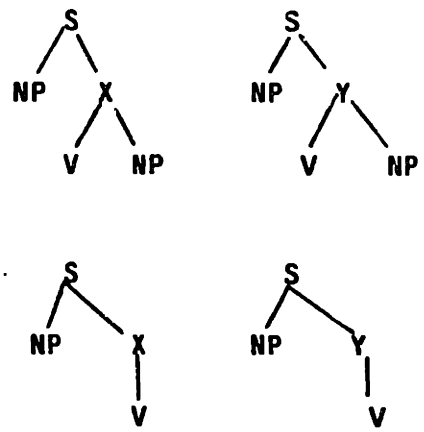


The entire development of induction in the finite state case can also be carried over wholesale to the induction of trees. The invariance relation of string concatenation must also be generalized, to that of tree concatenation. In the domain of trees, the analogue of k-tail suffixes are sub-trees. Instead of defining equivalent states as those that possess identical k-tails (hence "behave alike" given the behavioral criterion of string concatenation), the equivalent states of a tree automaton are those that have identical *tree residues*. Formally, a tree residue may be defined as follows:[58]

> Let S = a set of trees; t a particular tree. Then $\text{Residue}_t(S)$ = the set of trees u such that the subtree rooted at b is t, with the special symbol "$" replacing the subtree root node b.

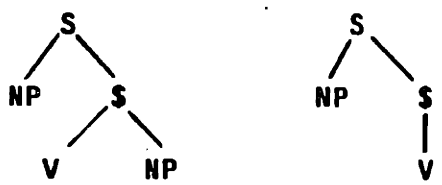
The dummy symbol "$" is just intended to serve as a placeholder.

For example, the trees below have identical tree residues at nodes X and Y:



Residues:

at X and Y:



58. From Levine [1981].

This fact allows us to draw the unsurprising conclusion that X and Y denote the equivalence class in a tree automaton for this system -- the node VP, of course.

The theorems of finite state induction theory also carry over to the case of tree automata induction, as Joshi and Levy [1977a] show. Just as all positive example *strings* of *length* less than 2n are sufficient to determine all the equivalence classes of a finite state machine with *n* states, all positive example *trees* of *depth* less than 2n are sufficient to fix the equivalence classes of a tree automaton with *n* states, equivalent to a grammar with *n* distinct non-terminals. Note that such an induction procedure assumes that the full bracketing of input strings is given or can be independently reconstructed.[59] Putting this major problem aside for the moment, the question of how much data is actually required for induction still remains. The number of (positive) trees of depth less than 2n for n of any size at all is rather substantial. For example, given even a simple $\bar{X}$ system, there are 4 major phrase categories -- Nouns, Verbs, Prepositions, and Adjectives -- plus particles, adverbs (as distinct from adjectives), complementizers, and modals. Thus all positive trees of depth less than 16 must be examined, a complexity that seems unrealistic given the human situation for language acquisition.[60] For example, Wexler and Culicover preclude their acquisition procedure from looking at sentences of this complexity for precisely reason that children cannot plausibly be expected to encounter examples of such complexity. Rather, they claim that examples of quite restricted depth are sufficient to cover the space of possible automata transitions, even though such a sample could not exhaustively span the space of possible surface strings. This distinction between a *covering* sample and an *exhaustive* sample is discussed in more detail below, for the case of finite state automata, and in Chapter 5, for the case of the Wexler-Culicover theory and Lparsifal. In any case, it appears as though "brute force" exhaustive induction procedures are too data-intensive to apply in the case of tree automata induction. For the method to become tractable, additional constraints must be supplied, such as those suggested by Wexler and Culicover, or those incorporated by Lparsifal.

To see how some of these ideas actually work in practice, let us apply the "state merger" approach to the induction of a very simple finite system, the auxiliary system of English. It should be emphasized from the outset that this application is intended to be illustrative. The demonstration does make some important points, however. There have been as many analyses of the auxiliary or inflection system of English in the linguistics literature -- a somewhat surprising fact, since, as is evident, the system is itself a restricted one that can generate only a finite set of strings. When analyzed from the standpoint of induction, however, it will be shown that all these proposals are about on a par.

---

59. In a more realistic setting, this bracketing might be deducible from cues such as pauses, intonation, or the like. Whether this is possible in practice, however, remains to be seen. See Morgan and Newport [1981] for preliminary psycholinguistic research on this topic.

60. More precisely, given the branching structure of English, trees of depth n correspond to sentences of length at least 2n, given the basic Governor-Governed element construction (P--NP, V--NP, etc.); e.g., "I know John knows Bill lies..."

First let us set the stage by outlining the major linguistic analyses of the English auxiliary system, given by Akmajian, Steele, and Wasow [1979 page 1]:

> The analysis of the auxiliary in English grammar has been, and continues to be a controversial area of reserach in recent theoretical lingusitics. Since the original transformational analysis of the English auxiliary given in Chomsky's *Syntactic Structures*, a number of modifications and challenges of that analysis have been presented...
> It seems fairly reasonable to view previous research on the auxiliary as dividing into two general proposals -- the phrase structure analysis (PS analysis) and the Main Verb analysis (MV analysis)... The PS analysis was originally proposed in *Syntactic Structures* with essentially the following phrase structure rule:
> (1) AUX-->Tense (Modal)(have + en)(be + ing)
>
> The facts to be accounted for can be stated quite simply: an English sentence can contain any combination of modal, perfective have, progressive be, and passive be, but when more than one of these is present, they must appear in the order given, and each of the elements of the sequence can appear at most once.
> [*Ibid.*, p. 17]

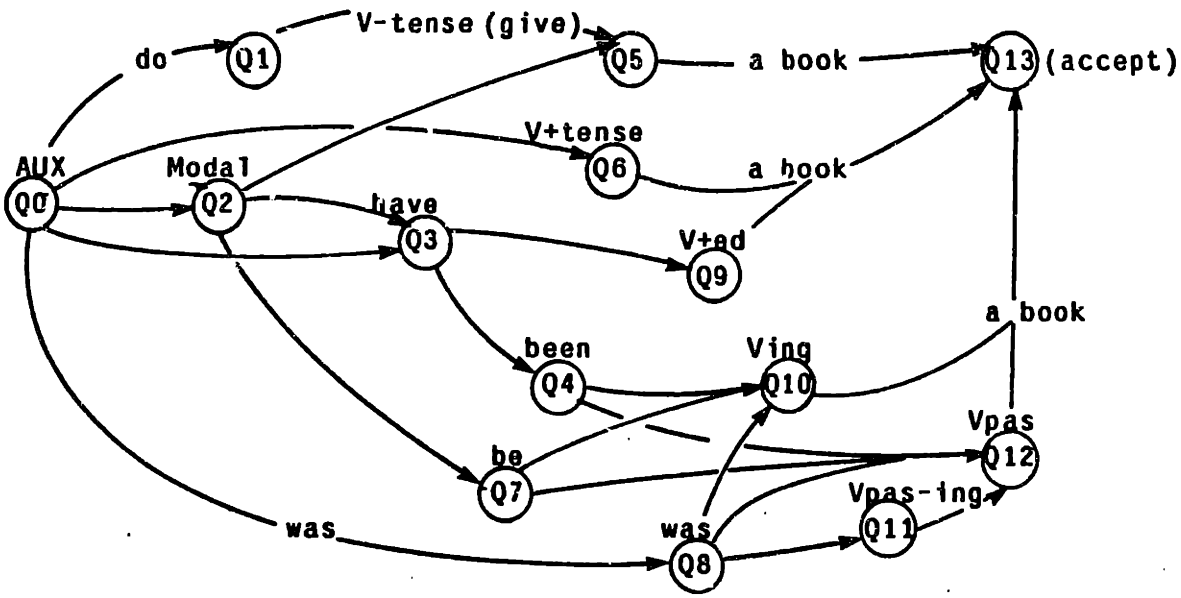
Formally, it is easy to see that Chomsky's phrase structure rule amounts to the specification of a very restricted finite state automaton (one that generates only a finite language); this is because a grammar for this language can be written without involving any self-embedding rules of the form $A \rightarrow \alpha A \beta$, α, β non-null, simply by replacing the schema above with appropriately labelled non-terminals to form a right-linear grammar. If we add a rule for *do* insertion, and attempt to incorporate passive/non-passive verb forms along with an object Noun Phrase, then the following sentences can be generated:

1. I could give a book.
2. I gave a book.
3. I did give a book.
4. I could have been giving a book.
5. I could be giving a book.
6. I could have given a book.
7. I have given a book.
8. I have been giving a book. (or books)
9. I have been given a book.
10. I could have been given a book.
11. I could be given a book.
12. I was giving a book.
13. I was given a book.
14. I was being given a book.
15. ?I could have been being given a book.
16. ?I will be being given a book.

17. ?I have been being given a book.

These examples have been taken from Chomsky [1975] *The Logical Structure of Linguistic Theory*, p. 231.) The reader is referred there for comments on the status of examples 15-17.

A finite-state system to generate this set of surface strings might look something like the following:

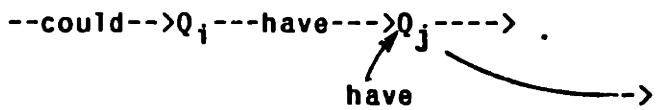


Including a non-accepting state (not included here for reasons of clarity), there are 15 states in all, and 25 "live" transition arcs (i.e., those not leading to the non-accepting state).[61] Since there are 15 states, it should suffice to provide an induction procedure with the value $n=15$ along with $S^+/<15$, i.e., all positive example sequences of length less than fifteen. Note that the bound on the number of states is nine, and therefore the expected *maximum* length of strings required to determine the equivalence classes of the resulting finite state automaton should be 2n-2, or 28. This means of course that the positive sample consists simply of all positive strings of the finite AUX sub-system.

One might also ask just how many data examples would be necessary for proper identification in this

61. Some of these arcs could be combined in an Augmented Transition Network -- e.g., the common NP arcs labelled "a book" could be merged by calling a common NP subroutine. These simplifications will not be pursued here.

case. Clearly, a necessary condition for successful induction is that every possible distinction made by the minimal automaton be exemplified by at least one example. That is, for every transition, $\delta(q_1, a)=q_j$ in the resulting finite automaton, there is some accepting sentence that causes the transition from q_i to q_j to be taken. Clearly, this is a necessary condition for the proper induction of an automaton from positive data. For if the collection of sample sentences is such that some transition $\delta(q_l,a)=q_k$ in the correct automaton is *never* taken, then there is no way that the acquisition procedure could deduce this; the procedure could not distinguish between two languages, one in which $\delta(q_l, a)$ leads to a valid state q_l, and one in which it leads to the non-accepting Thus the *minimum* number of positive data samples required is equal to the number of sentences sufficient to traverse all arcs of the resulting finite state machine -- a "covering" sample. It is apparent that a covering sample must always be at least as small as a complete sample. It is smaller in just those cases where the paths leading to a particular state Q_i is greater than 1 and the out-degree of transitions from that stateis greater than 1, so that the full multiplicative range of possible choices can be covered by a smaller number of examples. This is illustrated below for the case where two paths lead into and out of a state. Four possible paths are covered by three examples:



If all states have in-degree 1 or out-degree 1, then there is no difference between an exhaustive or a covering sample. In the case of the AUX system, a slight economy is possible because of examples such as the one above. As a result, a covering sample is slightly smaller than an exhaustive sample. The three sentences, "I have been given a book," and "I could have been being given a book," and "I will be being given a book," are all covered by other positive examples. (The first is covered by the two sentences, "I have given a book" and "I could have been given a book"; the second by "I have been being given a book" and "I could have given a book"; the last by "I will be given a book" and "I have been being giving a book".) The covering sample is thus of size 14.[62]

---

62. A covering sample will also be smaller if there are loops in the automaton. In the best case, a single positive example can cover all transitions. In the worst case, an automaton with *n* states, and with in-degree and out-degree 1 to all nodes, will require *n* samples for covering.

Note that the Wexler-Culicover Degree-2 theory advances constraints on a transformational system such that examples of bounded depth "cover" all possible transformational transitions. See Chapter 5 for additional discussion.

Let us now apply the state-merger induction procedure to the complete S^+ bounded sample. First, we calculate the "k-tails" for each of the strings in the sample, for k=28. (The Subject NP will be ignored here; also, the two tokens "a book" will be considered to form a single token.)

Token: tails

λ (All strings in S^+)

did	give a book
could	give a book, have given a book, have been given a book, have been giving a book, be giving a book, be given a book, have been being given a book be being given a book
have	given a book, been giving a book, been being given a book
was (am)	giving a book, given a book, being given a book
gave	a book

did give	a book
could give	a book
could have	given a book, been given a book, been giving a book, been being given a book
could be	giving a book, given a book, being given a book
have been	given a book, being given a book
have given	a book
was given	a book
was giving	a book
was being	given a book
gave a book	λ

could give a book	λ
could have given	a book
could have been	given a book, giving a book, being given a book
could be giving	a book
could be given	a book
could be being	given a book
was being given	a book
did give a book	λ
have given a book	λ
ave been given	a book
was given a book	λ
was giving a book	λ
have been being	given a book

could have given a book	λ
could have been given	a book
could have being giving	a book
could have been being	given a book
could be given a book	λ

could be being given	a book
could be giving a book	λ
have been given a book	λ
have been being given	a book
was being given a book	λ
could have been given a book	λ
could have been giving a book	λ
could be being given a book	λ
have been being given a book	λ
could have been being given	a book
could have been being given a book	λ

Note that there are 80 k-tails formed (63 plus 17 for the null string). Forming equivalence classes, we have the following states.

q_0= {all strings in S^+} (=initial state)
q_1= {giving a book}
q_2= {given a book}
q_3= {give a book}
q_4= {have given a book}
q_5= {have been given a book}
q_6= {have been giving a book}
q_7= {be giving a book}
q_8= {be given a book}
q_9= {have been being given a book}
q_{10}= {be being given a book}
q_{11}= {been given a book}
q_{12}= {been being given a book}
q_{13}= {being given a book}
q_{14}= {been giving a book}
q_{15}= {a book}
q_{16}= {λ} (accepting state)
q_{17}= non-accepting state

To calculate the transitions, recall that we form the transition $\delta(q_i, a) = q_j$ just in case the k-tail of *qa* is in the equivalence class q_j, for $q \in$ the q_i equivalence class. The transitions from the start state are as follows:

Initial state string x	input a	new state tail of xa	state
λ			
	do	give a book	3
	could	have given a book	4
	could	give a book	3
	could	have been giving a book	6
	could	have been given a book	5
	could	be giving a book	7
	could	be given a book	8
	could	have been given a book	9
	could	be being given a book	10
	have	given a book	2
	have	been given a book	5
	have	been being given a book	12
	gave	a book	15
	was	given a book	2
	was	given a book	1
	was	being given a book	13
	a book	null	17

(etc. -- all other transitions to state 17)

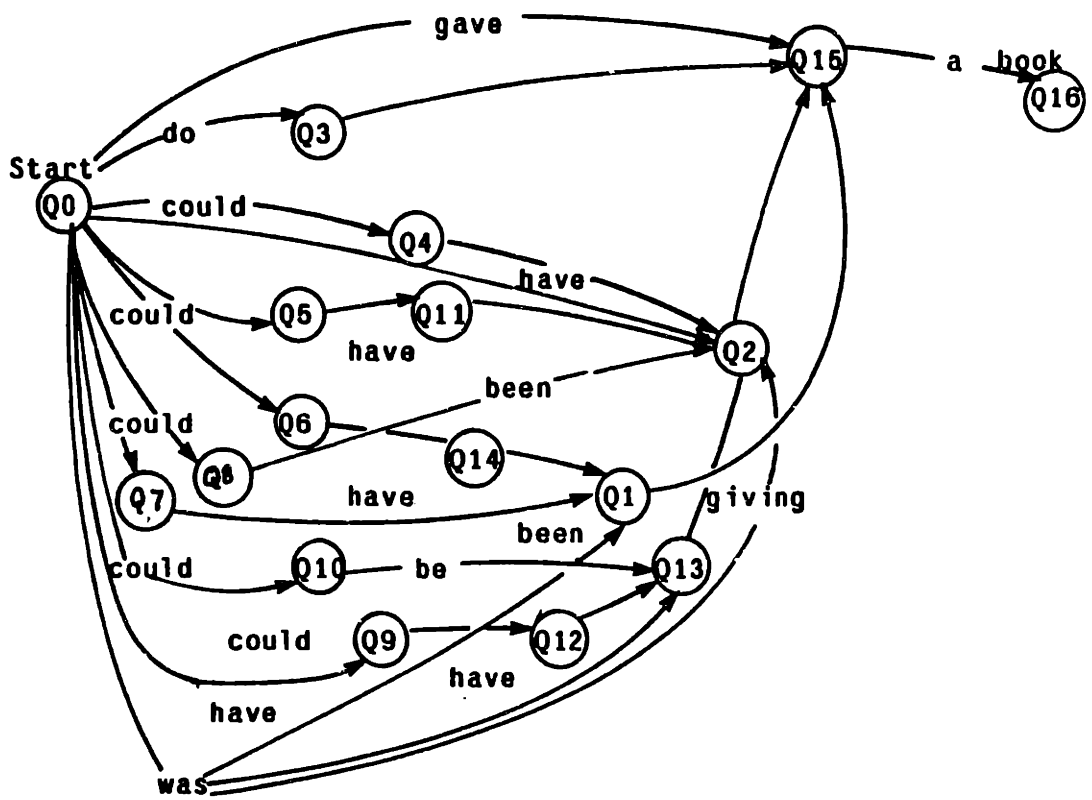
The remaining transitions are:

string	state	a	xa	k-tail	state
<u>could</u>					
give a book	3	give	a book		15
have given a book	4	have	given a book		2
have being given a book	5	have	being given a book		11
have been giving a book	6	have	been giving a book		14
be giving a book	7	be	giving a book		1
have been being given a book	9	have	been being given a book		12
be given a book	8	be	given a book		2
be being given a book	10	be	being given a book		13
<u>did</u>					
give a book	3	give	a book		15
<u>have</u>					
given a book	2	given	a book		15
<u>gave</u>					
a book	15	a book	λ		16
<u>was</u>					
given a book	2	given	a book		15
giving a book	1	giving	a book		15
being given abook	13	being	given a book		2
<u>could give</u>					
a book	15	a book	λ		16
<u>could have</u>					
given a book	2	given	a book		15
been given a book	11	been	given a book		2
been giving a book	14	been	giving a book		1
been being given a book	12	been	being given a book		13
<u>could be</u>					
giving a book	1	giving	a book		15
given a book	2	given	a book		15
being given a book	13	being	given a book		2
<u>did give</u>					
a book	15	a book	λ		16
<u>have given</u>					
a book	15	a book	λ		16

have been given a book	2	given	a book	15
gave a book	λ (final state)			
was given a book	15	a book	λ	16
was giving a book	15	a book	λ	16

(the remaining transitions are duplications of others, and will not be presented here)

The resulting automaton, in its unminimized form, looks like this:



The resulting automaton can be minimized via, e.g., the algorithm in Hopcroft and Ullman [1979]. (For example, several of the modal arcs can be collapsed together.) This step will be omitted here.

How complex is this inference procedure? At best, as we have seen, there are about n data samples for an automaton of size proportional to n. The k-tails must each be of length proportional to n -- at most, k times n. In the best case, there will be only one valid k-tail associated with any of the n data samples. (At worst, as we have seen, all the strings of length n could be in S^+, and thus there could be an exponential number of k-tails.) Calculating and collecting the n equivalence classes will require about n passes through a list k n long, or about n^2 steps in all. Calculating the transitions requires another pass thruogh the list of k-tails -- an additional kn steps. Minimization will take time n log n (Hopcroft and Ullman [1979]). Thus the total time will be on the order of n^2. In the worst case, as we have seen, the number of positive examples is proportional to 2^n, and in this case the number of steps the acquisition procedure takes will be at least this great.

In the worst case then, "brute force" finite state induction is exponential in the number of states of the finite state machine to be induced. This fact argues against the use of such an approach for actual acquisition. However, there are two replies to this criticism. First, though some constructed cases actually require that an exponential number of examples be examined, these cases may not actually arise in practice. For example, the induction of the AUX system does not require an exponential

number of steps. The number of data samples is of size proportional to n, the number of states in the minimal automaton for the language; the number of k-tails for any one of these samples is at worst of size k (as in the case of the tails for *could*), for a total complexity proportional to kn^2 for any one string. There are n strings, so the total complexity of the AUX induction is proportional to n^2. Note that using a covering sample instead will not reduce this number by any substantial amount.

There are induction procedures (based on method 2, state splitting,) that need examine only a polynomial number of examples. The basic idea has been suggested by Angluin. Instead of splitting equivalence classes based on a complete sample of sentences less than a certain length, one need only look at a subset of cases, namely a collection of sentences that puts the automaton through all possible transitions. Plainly, an acquisition procedure that works for all regular languages can use no fewer sentences than this. For suppose otherwise. Then there must be some transition that is not tested by the collection of sentences examined, for some regular language L, and machine M. Denote this transition by $\delta(q_i,a)=q_j$, and a string that prompts this transition by *xay*. Without loss of generality, assume that $\delta^*(q_j, y)= q_f$, that is, that *xay* is accepted by M But then, there is an automaton M^* that is just like M except that $\delta(q_i,a)=q_{na}$, i.e., *xay* is rejected. If the acquisition procedure cannot test string *xay* then it cannot distinguish these two machines, a contradiction. A covering sample, then, is *necessary* for correct induction. Is it also sufficient? Intuitively, the answer is yes: if one is attempting to split automaton states, then no more states need be distinguished than those split by transitions of the machine. (For a detailed proof of this intuition, see Angluin [1980].)

Does the order in which examples are presented matter? So far it has been assumed that the entire sample S^+ is given to the acquisition procedure all at once, an instantaneous model. What happens if only one example from S^+ is examined at a time? Given the k-tail framework, if the order of examples is to matter then there must be some sequence of examples where a k-tail merger made on the basis of some initial segment of examples is later disconfirmed. For example, consider the following pair of examples:

have given the book
was given the book.

Given just these two cases, a k-tail acquisition procedure must conclude that *have* and *was* are in the same equivalence class. But this conclusion is overturned by the subsequent examples:

have been given the book. *was been given the book.

alternatively: was giving the book. (*have giving the book)

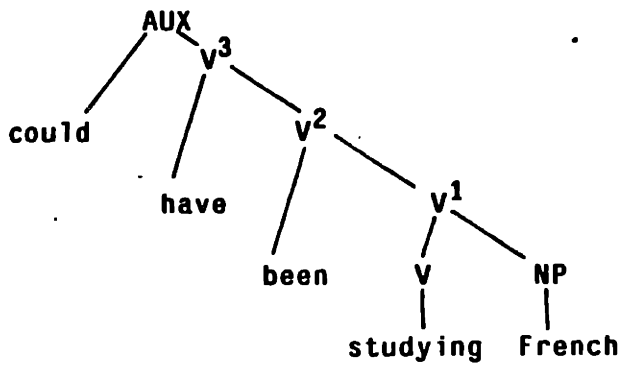
These two cases (the second a case of indirect negative evidence) force the acquisition to put *have* and *was* in difference classes, because the k-tails are no longer the same. In short then, if the acquisition

procedure is permitted to go back and split classes than have previously been merged. then acquisition can still proceed as before. Alternatively, if one adopts a class-splitting procedure to begin with (as in Angluin's polynomial time algorithm, mentioned above), then one need only split up classes as evidence is encountered to prompt such splits. For example, given the initial two examples above, a class-splitting procedure would simply place *have* and *was* in the same class, then split them when the examples have been given the book--was giving the book come along. It appears then as though the class-splitting procedure is somewhat more natural in a context where examples are presented one at a time. In any case, it does not seem that the order in which examples are presented adds any ne w constraint to the acquisition process, at least in this simple case of the induction of the AUX system.

As mentioned earlier, other analyses of the English Auxiliary system have been proposed. Do these make a difference for acquisition? For example, Akmajian, Steele, and Wasow [1979] have suggested that the AUX system is produced via simple phrase structure rules like the following:

AUX-->{Tense *do*, modal}
V^n-->[+V +AUX] V^{n-1}

This would produce branching structures like the following:



This grammar overgenerates, of course, since the order of *have* and *be* has not yet been encoded. Akmajian, Steele, and Wasow propose to capture the ordering restrictions via subcategorization frames. Perfective *have* will demand a V^2 complement, progressive *be* a V^1 complement, and passive *be* "must be immediately followed by a main verb". [1979 page 21]. Thus the order of non-terminals as 3-2-1 encodes the relative order of *have* and *be*.

From the point of view of inference, there is little difference between this proposal and the earlier one. The Verb subcategorization analysis forms a right-branching structure -- again, generating only a regular language. The number of non-terminals, hence the number of states, is the same in both the traditional *Syntactic Structures* analysis and in the ASW theory. That this should be so is not

surprising. Somehow, any grammar for the AUX system must encode the descriptive fact that the elements modal, *have*, and *be*, are ordered with respect to each other. This information must be acquired if it cannot be deduced from other (perhaps universal) constraints. In other words, any theory for AUX must generate the right *language*. If the language is regular (and finite), then it is not surprising that there are a variety of proposals that can represent a minimal finite state automaton for that language.[63] Therefore, any proposal that does not reduce the grammar of the AUX system to other, more general constraints is on roughly the same acquisition ground as any other proposal: it demands the induction of a finite state machine of about the same size as that demanded by any other descriptively adequate system that has been suggested in the literature.[64]

Finally, Baker [1981] has recently advanced a variant of the Akmajian, Steele, and Wasow approach that attempts to do away with a three-level analysis of Verb Phrases. Instead, it generates all Verbs at a V^1 level and introduces V^1 as a possible sub-constituent of Modals:

S-->NP {M^1 or V^1}
M^1-->Modal V^1
V^1-->V V^1 or V NP.

To avoid overgeneration, Baker again resorts to subcategorization restrictions; he suggests that lexical entry for *have* contain the insertion frame context <___V^1 en>, that for *be* contain the context restriction <___V^1 ing>, that for do <___V stem>.

Plainly, Baker's rule system is also right-branching. There are some differences between it and the two previous theories, however. Evidently there are fewer non-terminals in this account than in either the ASW or *Syntactic Structures* theories, since there is only one phrase of type V. This would seem to be an advantage. Baker's suggestion is also compatible with a more radical proposal that will be described in more detail in the next section of this chapter, namely, the elimination of all phrasal category names. Observe that all of Baker's rules are of the form Head-Complement, where Head = Modal or V. This form suggests that constraints on ordering may indeed be naturally captured via restrictions on allowable complements, and these perhaps may be derivable from other, more general constraints.

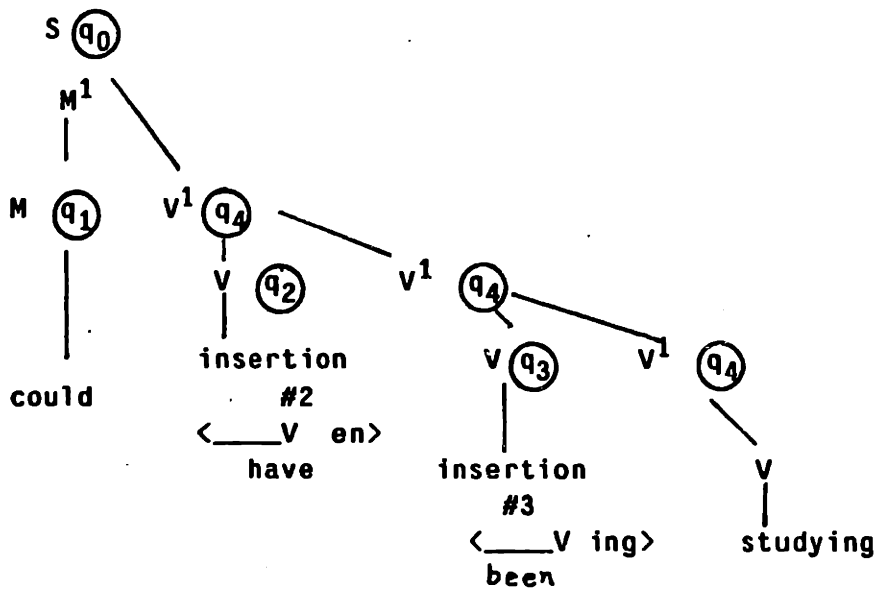
---

63. Baker [1981] remarks that Emonds [1969] contains a proposal to deduce the ordering of elements of AUX from general constraints on verbs. This approach seems more plausible given the current Government-Binding theory, and we will return to it below.

64. As ASW observe, there are other advantages to their analysis that go beyond an account of the order of elements in the auxiliary. For example, the rules are designed to provide a natural explanation of what the natural units should be for movement rules like Verb fronting. Under their proposal, the phrases *study French*, *been studying French*, *have been studying French*, and *could have been studying French* are complete phrases of type V^i, hence can be moved as complete units. This explanation is not immediately available to a system that generates a "flat" auxiliary. However, it may be possible to reproduce this effect by combining the government-under -adjacency constraint with the Hornstein-Weinberg notion of reconstruction; see below for additional discussion.

This topic will be explored in the next section. On the other hand, the information content that is captured in other systems by using non-terminal labels still needs to be acquired in this system -- only that information is now associated with lexical items. This is to be expected in a system that attempts to eliminate phrase structure categories: to remain descriptively adequate, the information formerly encoded via non-terminal labels must be encoded somewhere. Thus each of the lexical restriction frames, [be: <___V^1 ing>], etc., must be learned. The advantage to this approach is that this information might also be needed for other phrase types besides AUX -- for example, for ordinary Verb Phrases.

How much extra information is encoded in the lexical entries? As shown by Peters and Ritchie [1969] context-sensitive insertion frames of this kind can be simulated by a finite-state frontier-to-root tree automaton that "checks" whether the context conditions are met.[65] The induction of this automaton replaces the induction of the finite state machine of other analyses.



If the trees defined by the base grammar are just right or left branching, then the resulting automaton is just an ordinary finite-state automaton, as noted earlier. But then, the number of states in this automaton must be at least as great as the number in the minimal (reduced) automaton for the AUX language, i.e., the same number or larger than that for the first two proposals above; otherwise, there would be a finite automaton smaller than the minimal automaton for this language, a contradiction. Thus the total number of finite automaton states that must be induced in Baker's approach must be

65. See also Thatcher [1973].

the same (or greater) than the number involved in the *Syntactic Structures* or Akmajian, Steele, and Wasow approaches, despite appearances to the contrary. In fact, this conjecture is borne out. If one augments Baker's system so as to include *do* insertion, passive *be*, and NP, start, and reject states, the the number of states required is the same as in the generative approach:

Start, Accepting, Non-Accepting states= 3

```
NP:          [a book]

V-tense:     [____NP]
V+tense:     [____NP]

do:          [____Vmainstem]
             (required to block "I do have taken...")
Vmainstem:   [____NP]

be:          [____Ving]
have:        [____Ven]
modal:       [____V-tense]

passive be

been:        [____Ven-passive]
was:         [____Ving-passive]
was:         [____Ven-passsive]
being:       [____Ven-passive]
```

However, it is important to stress again that there is an advantage to the elimination of non-terminals if one can derive these general context-sensitive insertion frames from general principles. It is the overall acquisition complexity of a grammar that counts, not just the complexity of one portion of it. This being so, then the overall grammar is simplified if general principles can be used to explain the acquisition of several of its components, instead of having specific principles of each separate component.

Finite-state induction and Lparsifal

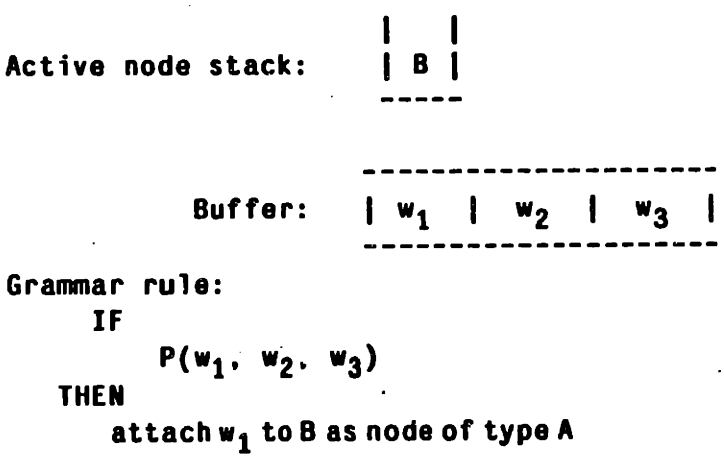
The way in which Lparsifal acquires the AUX system from positive examples is perhaps closest in spirit to the acquisition of context-sensitive insertion frames, as in Baker's model. This is true for two reasons: (1) the tree shape induced is the same, roughly, an X^0-X-complement structure; and (2) the pattern action grammar rules are just local context sensitive insertion rules.

IF
<predicate true of the input buffer and stack>
THEN
<rule action>

The predicate patterns are just context-sensitive feature tests as used by Baker: the features =/- Tense, +Verb, etc., are those typically used by Parsifal. The predications are also strictly local, in the sense of Joshi and Levy [1977b], the same sense that Baker intends. Specifically, a predicate is *local* just in case it is a Boolean combination of *proper analysis predicates*, where a proper analysis predicate is defined inductively as follows:[66]

(i) if t=0 (null tree), then P(t)=Ø;
(ii) if t= root A dominating subtrees
$t_0 ... t_n$, then $P(t)=\{A\} \cup P(t_0) ... P(t_n)$ -- i.e., the union of
A with the concatenation of the proper analyses of the subtrees below A.)

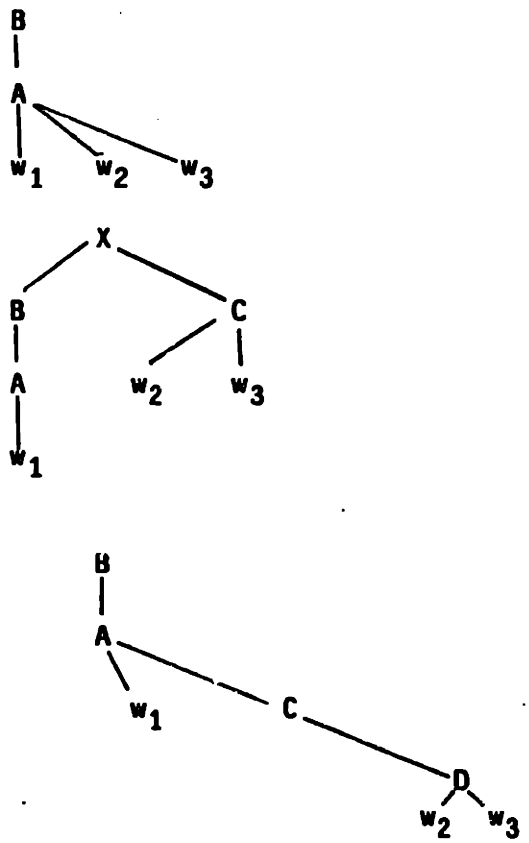
Intuitively, a proper analysis predicate specifies a horizontal context "slice" through a tree that passes through the node being re-written (node A), touching one representative node from each subtree in the horizontal slice. It should be apparent than any Parsifal grammar rule predicate that contains only lexical tokens in its input buffer meets this definition of local:



This grammar rule corresponds to a re-writing rule of the form, B-->A/Φ____γ, where Φ=predicate defined on active node stack; γ= predicate defined on buffer. These predicates are local, since nodes in the active node stack must lie in sub-trees to left of node A, and the trees to which w_2 and w_3 will be

66. This notion of a local re-write rule was apparently first suggested in Chomsky [1965], who called it a "local transformation."

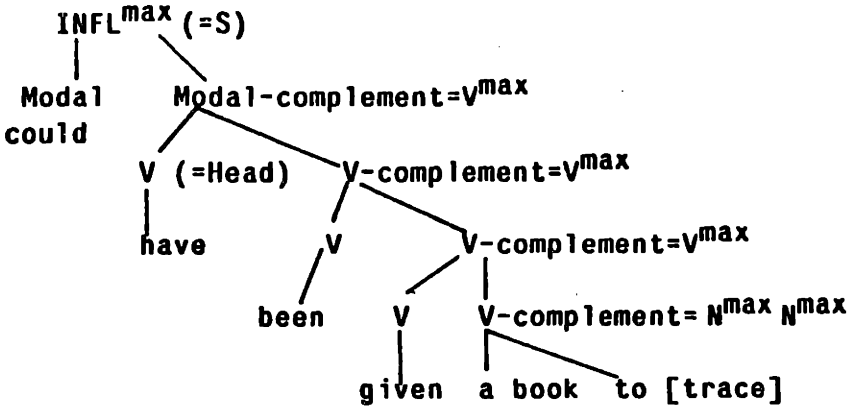
attached must lie to the right of A:



From this perspective, all Lparsifal acquires is local insertion frame rules. Lparsifal also follows Baker's approach in that there is just one kind of "phrase structure" schema that is used to develop the AUX system, of the form:

X^{max} --> X X-Complement

Obviously, the order X--X-Complement must be acquired first. The final attained system is designed to look like this:



The structure is right-branching, just as in Baker's model, and there are no "levels" of the form V^3, V^2, V^1, as in Akmajian, Steele, and Wasow. The aim is to avoid the use of level numbers, which, after all, simply denote different non-terminal names, hence are used simply to distinguish machine equivalence classes. But these classes can just as well be defined via context-sensitive insertion frames.

Given this framework one can now almost do away with phrase structure rules altogether, at least in this simple case. Suppose that one takes the notion of government as a primitive for Lparsifal, along with that of string adjacency. (Observe that the string adjacency primitive is tacitly assumed in the way that the buffer works: an item can be next to another item.) Government may be defined as follows:

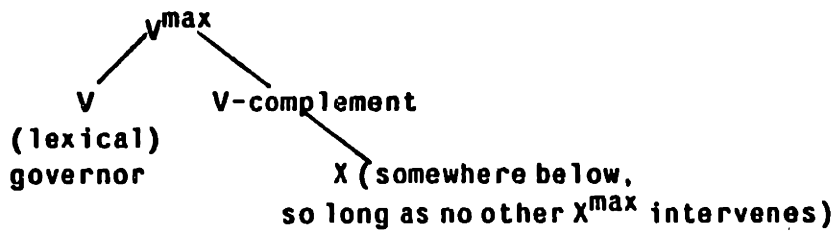
> In a structure of the form, $[\beta \ldots \gamma \ldots \alpha \ldots \gamma]$, where
> (i) $\alpha = X^0$
> (ii) where φ is a maximal projection, if φ dominates γ then
> φ dominates α.
> (iii) α is an immediate constituent of β
>
> α governs γ.
>
> [Chomsky, 1981, page 163]

In the usual case, $b=\overline{a}$, but not always, as in, perhaps, *[for [John to go away] would be terrible]*, where *for*$=\alpha$, but $\beta=S$.[67] Government is also closely related to subcategorization and case marking. For example, if α case marks a constituent, as in, *to the store*, then α governs that constituent. The intent of clause (ii) of the definition is to ensure that a maximal projection does not intervene between α and the element γ that it governs. In other words, the governor and its governed element must be in the same

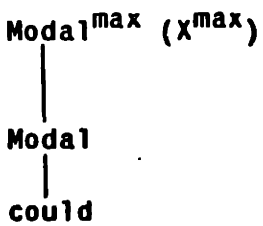
---

67. Though, as noted by Chomsky, one might get around this by assuming that *for* as COMP is the Head of S. This proposal actually fits quite well into the Lparsifal model as we saw earlier in Chapter 2.

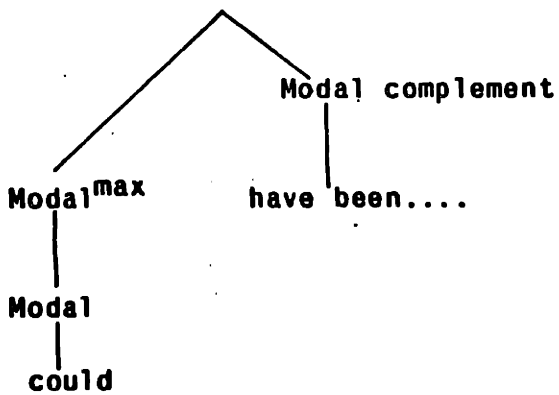
maximal projection domain. If we know that V governs an element X, and that X lies to the right of V, then the relationship between V and X must be as follows:



In particular this is true if X is a phrase that is a sister of V. Hence V always governs its own complement phrase. Given this fact, the structure of the AUX system can be derived from that of government. For suppose it is known that the Modal *could* governs its "complement" *have been given a book*. The modal is projected to form an X^{max} category of some sort:

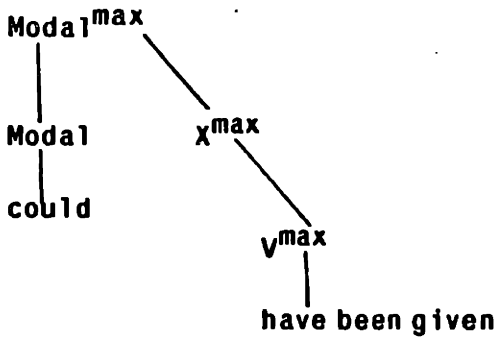


How is the complement phrase to be attached? it cannot be attached *above* the maximal projection of modal, for then otherwise modal would not govern its complement:

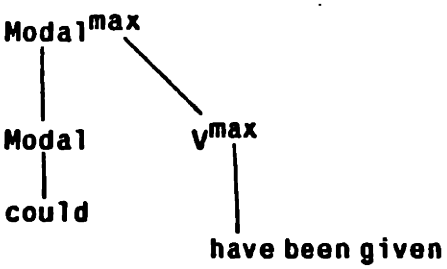


What choices are left for the attachment of the Complement structure? Suppose that the Complements are always the maximal projection of some category. Then the only place for attachment of the Modal complement is as an immediate constituent of the same maximal projection

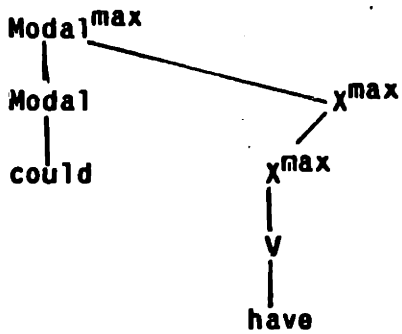
domain as the Modal. For otherwise, a configuration results where there is a maximal projection φ such that φ dominats γ= the Modal complement, but φ does not dominate the Modal:



The only possibility left is:

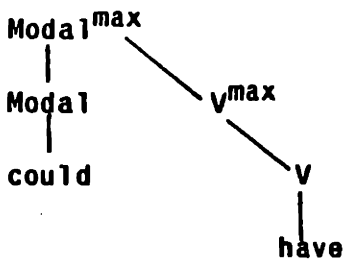


Continuing this line of reasoning, suppose it is known that *have* governs its complement, *been given a book*. Then it too must correspond to a fragment of phrase structure that obeys the basic government configuration. The X^0 lexical items projects to an X^{max}, as before:

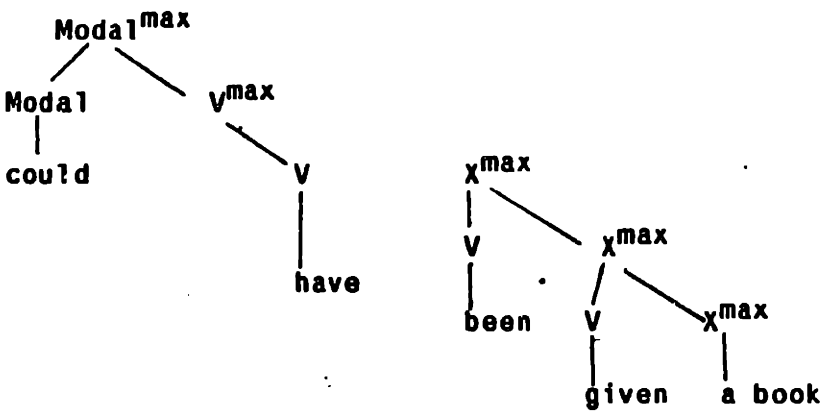


But then, the two X^{max}'s have to be identified, or otherwise there is a violation of government, as

before. Thus the structure must be:



Similarly, the structure of the complement (governed) constituent of *have* can be recovered. The final result is:



We need no "rules" of phrase structure expansion to determine the required underlying representation in this case. To this extent the context-free re-write rules that "describe" the arrangements of constituents in surface structures to which no movement rules have been applied are epiphenomenal; they are parasitic on the "deeper" notions of adjacency and government.[68]

So Lparsifal really needs two kinds of information in order to fix rules to handle the AUX system: (1) context-sensitive conditions for insertion of terminal items and (2) knowledge of whether one item can govern another or not. In simple cases, information about government amounts to knowing whether the governed constituent is part of the complement structure of the governor -- roughly, subcategorization information, as Chomsky has noted. That is, part of the meaning of a Verb such as *take* is its argument structure: *take* -- something. This fact about what *take* "means" is assumed to be

68. This leaves open the possibility that Government is itself derived from other terms of the grammar.

derivable from extra-linguistic information ("situations", say), just as in the Wexler-Culicover theory, base structures representing predicate-argument relations are assumed to be available to the acquisition model. The current assumption is weaker than the Wexler-Culicover theory, however. Where the Wexler-Culicover model assumes that the full deep *structures* can be recovered from "context," the current model demands only that the unordered argument or thematic role requirements be so recoverable.[69] For instance, in the case of *take*, all that would be assumed known is that *take* assigns the thematic role of Patient, as the thing taken. This thematic information is "projected" through the constraints of lingusitic structure: namely, an assumed correspondence between NPs and things (denoting objects in the world), and the default adjacency requirement on case assignment, hence assignment of thematic roles.

Let us see exactly how these principles work in the context of several examples drawn from Chapter 2. We will just review the examples here; for details, see Chapter 2.

Example 1. John has the book.

Step 1: Assume that *John* is parsed correctly as an NP. The grammar rules to do this will have been acquired by exposure to simple declarative sentences (see Chapter 2). Also assume that the basic $\bar{X}$ order of English sentences has been previously determined as Specifier-Head-Complement = NP INFL VP. The parse begins by assuming that some X^{max} (=S) will eventually be found. The packet Parse-specifier is entered, and the NP *John* is parsed normally, leaving the parser in the state shown below:

```
packet: Parse-Head-INFL
 S          -----------
 T          |  X-max  | (INFL)
 A          |  Spec   |
 C          |  NP     |
 K          -----------

           ----------------------
buffer:    | has | the |  book  |
           ----------------------
```

What happens next? There are basically two options. First, one could assume that *takes* can be

---

69. Wexler and Culicover [1980] point out that they need an independent theory of how these base structures are built up, and that something like $\bar{X}$ theory would be of immense help in this regard.

recognized as a Verb plus some kind of Tense Operator, and that INFL cannot take a verb as its Head. Since the verb cannot be attached directly to INFL, the operator is *switched* into proper position.

Alternatively, suppose that Tense is not recognized as such, but that *has* is known as a V. Then since the category INFL is not developed, there is nothing to *prevent* it from being attached as the Head of INFL; in effect, the system collapses the categories INFL and VP together as a single "operator", since it has no basis on which to separate them. Observe that this has the effect of making V^{max} *temporarily* the Head of S, a conclusion that will be withdrawn as soon as the INFL node is developed.

The Parse-complement packet is now entered. If option 1 has been selected, then the +V item *has* is assumed to prompt the creation of a V^{max}; this node is created and placed on the active node stack. Under the second option, assumption that a Tense operator is unknown, *has* can be assumed to take some kind of X^{max} complement, so an X^{max} node is created and placed on the active node stack.

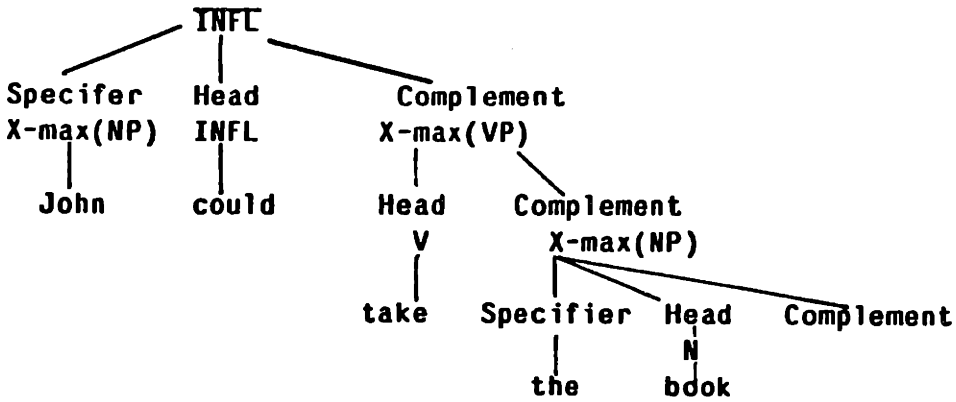
The Specifier, Head, and Complement of this new X^{max} are now parsed. Eventually, under either option the NP *the book* will be attached as the complement of *have*. (See Chapter 2 for details.) The acquisition procedure has acquired the insertion frame for the complement of *have*.

Now consider a more complicated example, *John could take the book*. The NP *John* is parsed as before, leaving the parser in the following state: The parser will now attempt to parse the Head of INFL. There is a difference this time, however: *could* is not a Verb, and is in fact not currently marked for any feature. Thus there is nothing to prevent it from being attached as the Head of INFL.

Next, the complement of INFL is parsed, only this time the type of complement phrase is different, being the VP *take the book*. An X^{max} must be formed and placed on the active node stack, but what type of node should this be? Again, there are two possible lines of inference, but they both lead to the same result. First, suppose that it is known that every sentence must be of the form NP INFL VP. Then the complement must be a VP ($= V^{max}$). Hence the Verb *take* must be the Head, since it is the only possible Head element. Alternatively, suppose that a node of type X is created, while remaining neutral as to the identity of the category. Assuming that is known that *take* assigns a thematic role to its argument *the book*, and assuming now that government is restricted so that the governor α is always the Head of the maximal projection containing both the governor and the governed constituent, then *take* must be the Head of this X-phrase.

In either structure, *take* would fail to govern *the book* as required (under the restricted definition of government).

The new X-phrase (a Verb Phrase) then prompts the parse of a Complement (as in a simple declarative sentence), and the NP is analyzed as before. We arrive at the following parse tree:



Now consider, *John has taken the book.* This will be analyzed roughly like the preceding sentence, but with one exception. If we assume that INFL cannot dominate a V node, and that *has* is broken down as *have+Tense operator,* then the Tense Operator will be *switched* into proper position and attached under INFL. The grammar rule for this will be:

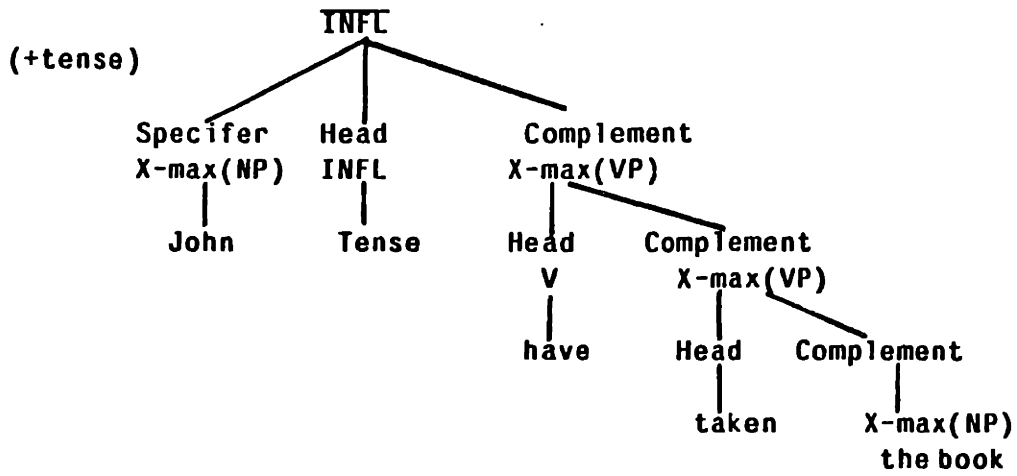
```
IF    [current cyclic node is null]
      [current active node is INFL-max]
      [active packet is Parse-Head]
      [buffer is: | have +V | Tense Opr | take +V |]

THEN
      switch


IF    [current cyclic node is null]
      [current active node is INFL-max]
      [active packet is Parse-Head]
      [buffer is: | Tense Opr | have +V | take +V |]

THEN
      attach
```

Next, the Complement of INFL will be analyzed as before. The final tree will look like this:



(This analysis ignored the tense operator attached to *take*.)

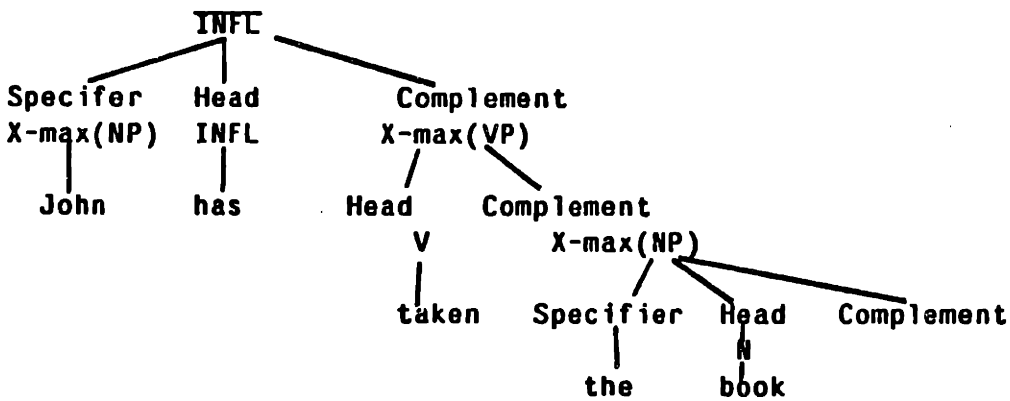
Suppose, though, that we assume a conflation between INFL and V projections, as in the second of the two options discussed above. Then *has* can be attached to the INFL node, just like *could*. The grammar rule and final tree under this analysis is as pictured:

```
grammar rule:

 IF     [current cyclic node is null]
        [current active node is INFL-max]
        [active packet is Parse-Head]
        [buffer is: | have +tense, +V | taken +V +tense | the +N |]

   THEN
            attach

tree:
```



The grammar rule to attach *has* is much like that to attach *could*; the same packet is active in both cases, and the active node stack contains the same element. By definition therefore, the state of the parser *before* it decides what to do with *could* or *have* is the same in either case. However, recall the conditions dictating grammar rule merger from Chapter 2:

> (1) (Non-terminal merger rule)
> If there are two rules of the form $A \rightarrow \alpha / \Phi\_\_\_\gamma$, $A \rightarrow \beta / \Psi\_\_\_\Omega$, and $\Phi = \Psi$, so that the left-hand context, hence parser state, is the same in both cases, and $\gamma = \Omega$, so that right-hand context, hence look-ahead buffer, is the same in both cases, then α and β denote the same non-terminal element (are in the same equivalence class). In other words, the class $\{\alpha, \beta\}$ is generalized by the merger of α and β, the unification of their feature sets. (This is implemented simply via set intersection.)
>
> (2) (Pattern merger rule)
> If there are two rules of the form $A \rightarrow \alpha / \Phi\_\_\_\gamma$, $A \rightarrow \beta / \Psi\_\_\_\Omega$, with $\alpha = \beta$, and $\Phi = \Psi$, then the *rule* $A \rightarrow \alpha$ is generalized by unification over the look-ahead buffer patterns, γ, Ω.

These two conditions follow the dictates of the automata induction theory developed earlier: two tokens are in the same equivalence class just in case, starting from the same machine state, they lead to identical new states. The condition on identity of left-hand contexts and packet names (the parser

state) is just a translation into Lparsifal machine terms of the condition that one start from the same machine state; the restriction that either the look-ahead buffer be the same in both cases or that the node being reduced back to an A be the same in both cases is just the condition that the new state after a transition be the same in both cases. (Formally, given two initial states, q_i and q_j, then for $\delta(q_i, a)$ to be equal to $\delta(q_j, b)$, either (i) q_i and q_j are the same state, and $a=b$ -- the pattern merger condition; or else (ii) q_i and q_j denote the same machine state, $a \neq b$, but $\delta(q_i,a) = \delta(q_j,b)$ -- the non-terminal merger condition.)

In the case at hand, the rules for *could* and *have* have different look-ahead patterns: *could* has the tenseless form *take* in the second buffer position, while *have* has the tensed form *taken*. In fact, this is no more than k-tail induction; since the look-ahead buffer suffixes of the two parser states are different, the two rules are kept distinct. In contrast, consider what happens with the sentence, *I will take the book*. The grammar rule that the acquisition procedure forms to attach *will* is exactly like that used to attach *could*, but for the difference in form between *could* and *will*. Therefore, these rules meet merger condition 1, and the tokens *will* and *could* are placed in the same equivalence class. For an example where merger condition two applies, consider the sentences, *John could buy the book* and *John could take the candy*. Here, the acquisition procedure will build two rules, A-->could/Φ___buy the book, and A-->could/Φ___take the book, thus prompting a pattern merger to, e.g., A-->took/Φ___+V -tenseless the book.[70]

Summarizing so far, the following insertion frames have been acquired:

```
(Modals) {will, could,...}  [will, could ][take +V -tense][the]
{have}                      [ have ][taken +V +tense +en][the]
{take, taken}               [ take, taken][ NP ]
NP                          [ a book, the book]
```

The target set of insertion frames is:

---

70. Of course, this approach in no way *explains* the insertion frame for *have*: [___+V +ed]. Things would have been just as simple if the frame were: [___+V -tense], just like *could*. One speculation: suppose that NPs and tensed clauses are alike, in that they both pick out an individual object (in the model theoretic sense) from a possible domain of individuals; e.g., "the book" denotes an individual, and "taken a book" or "is sick" can also be (partially) evaluated; but "take a book" cannot be, since the modality is unexpressed. Then the fact that *have* and *be* require tensed Complements as Auxiliary verbs would follow from their subcategorization frames as Verbs. Similarly, since modals such as *will* and *could* never take NP complements, they cannot take the Verbal analogue of NPs, tensed Verb Phrases.

```
1. NP:          [a book]

2. V-tense:     [____NP]
3. V+tense:     [____NP]

4. do:          [____Vmainstem]
              (required to block "I do have taken...")
5. Vmainstem:   [____NP]

6. be:          [____Ving]
7. have:        [____Ven]
8. modal:       [____V-tense]

9. passive be

10. been:        [____Ven-passive]
11. was:         [____Ving-passive]
12. was:         [____Ven-passsive]
13. being:       [____Ven-passive]
```

The acquired frames correspond to the target frames 1,2,3,8, and 7. Note that the Lparsifal buffer patterns, e.g., [modal][V +en] are just a variant notation for insertion frames, e.g., V-tense: [___NP]; the lexical entry V-tense just becomes the first item in the buffer, the element being attached to the parse tree.

Further positive examples will fill in the additional insertion frames required for the Aux system. For instance, consider the sentence pair, *I could take the books* and *I could give the books*; by merger rule 2, the attachment rule for *could* is generalized to [modal][+V -tense][the]. The example *I could give books* invokes merger rule 2 a final time to yield the desired pattern, [Modal][+V -tense]. Similarly, the examples, *I am giving books, I am taking books, I am seeing John* yields the right insertion frame for be, [___Ving].

Acquiring the other insertion frames is slightly more complicated, however. Take the insertion frame for *do*. Examples such as, *I did take the books, I could take the books* will be encountered. This example will place *did* in the same category as the Modals for the purposes of the Aux system. Of course, the example *I could have taken a book* immediately distinguishes the two:

```
could:      [ could ][ have +V -tense][ taken +V +tense +ed]
did:        [ did  ][ take +V -tense][ NP]
```

Examples such as these suggest that while the class merger of lexical items under Condition 1 above is on the right track, it requires some modification. Condition 1 really attempts to form equivalence classes of lexical entries. As such, it seems reasonable to expect that crude categorizations of lexical entries might be replaced by more refined ones. For instance, the Keil studies indicate that categorization abilities undergo an incremental process of foliation and refinement, as discussed in the previous section of this chapter. If this is so, then it seems natural to add the ability to split existing lexical class categorizations at any time, given positive evidence. Thus, while *could* and *did* would be placed in the same lexical class via exemplars such as *I did take/I could take*, they would be split by examples such as the pair above. The crucial minimal pair could come either before or after the *I did take/I could take* pair, so in fact depending on the chance of external events, *did* and *could* might never be merged at any time. Note that lexical class splitting is to be distinguished here from rule merger under Condition 2.

Another example of the same phenomenon is provided by the pair, *I was given a book/ I have given a book*. The "tails" of both examples are identical, as are their left-hand parsing contexts at the time that *was* and *have* must be attached. This identity would seem to prompt an (incorrect) rule merger under Condition 1. Here, however, there is a way to prevent the merger of *was* and *have*. Suppose (as outlined in Chapter 2) that a passive rule is acquired via the analysis of examples such as, *John was kissed*. The grammar rule rule for the passive will look something like this:

```
IF  [current cyclic node is INFL-max +be]
    [current active node is V-max +V +Arg]
    [current packet is Parse Complement-V
    [buffer is:  [ -N ][ ][ ] or [ # ][ ][ ] ]
THEN
          drop trace
```

This rule has two effects:

1. A trace is dropped into the first position of the input buffer.
2. The current active node is annotated with the name of the rule just executed.

Action number 2, it will be recalled, is a concomitant of every grammar rule. Its effect in this case is to label the current active node, *was*, with a distinguishing label, the name of the just-acquired rule. This label is actually a "+passive" marker, triggered by the passive morphology. Given this feature, and the assumption that some form of the passive rule has been acquired, then merger of *was* and *have* is in

fact blocked. This is because *was given* will trigger the passive rule, whereas *have given* will not. The labelling of the INFL node makes the left-hand context of the the *was given* sentence different from that of *have given*; therefore, the rules are not merged.

In sum, the Lparsifal acquisition procedure mimicks the finite-state "k-tail" method in its analysis of the Auxiliary system, as applied to the induction of the context-sensitive insertion frames (equivalently, states of a tree automaton). The limit on the look-ahead buffer to three constituents implies that the "tail" that can be examine is limited -- at most three lexical items (in the case of linear strings, as in the Aux system), or, more generally, three constituents. Interestingly, as the reader may verify, even though the AUX system requires on the order of a dozen or so states, the actual length of example suffixes required to distinguish between any pair of states is three or less. Note that this condition is implied by Marcus' hypothesis that a 3-cell look-ahead bound suffices for the deterministic parsing of English. For suppose that the Aux system is not learnable via a 3-cell k-tail method. Then there must be two distinct non-terminals, A and B, such that A and B cannot be distinguished via left- or 3-cell right-contexts. But then the system cannot be parsed deterministically by the Marcus parser, using a 3-cell look-ahead buffer. This is because by the assumption of non-learnability, then for all rules re-writing A and B, $A \rightarrow \alpha / \Phi \_\_ \gamma$ and $B \rightarrow \beta / \Psi \_\_ \Omega$, $\alpha = \beta$, $\Phi = \Psi$, and $\gamma = \Omega$. Therefore, we cannot tell whether to reduce α as either A or B, and deterministic parsing fails. Hence, ¬3-tail learnable implies ¬Marcus 3-cell parsable. Forming the contrapositive, we have that Marcus 3-cell parsable} 3-tail learnable.

We have established the following:

Theorem 3.1 Let G be a grammatical system that is k-cell deterministically parsable. Then G is k-tail learnable via positive examples, using Revised Condition 1 and Condition 2.

3.4.3 Government and Precedence Parsing

As several sections of this chapter have demonstrated, knowledge of language need not be *explicitly* represented as re-write rules, traditionally construed. This is an important point that distinguishes the acquisition of knowledge of language from the use of that knowledge. Of course, knowledge and its use are inter-related; what this thesis has presented is an implemented computational model that shows specifically how the acquisition of rules to parse English sentences interacts with the acquisition of knowledge of English syntax. Evidently, knowledge of syntax and ability to parse are not one and the same, however, since the converse of the the theorem just cited above is false. There are languages that are k-tail learnable, but not deterministically parsable. To take an example presented earlier, we need only consider an ambiguous grammar that is k-tail learnable, e.g., VP⇒V NP (PP); NP⊂N (PP). Still, this is an unremarkable fact. It is not surprising that it is possible to easily acquire a

representation of a certain sort of knowledge, and yet what *follows* from that knowledge may not be computationally tractable. A representation of the simple rule system given just above is easy to learn: all that is required is the order Head-Complement, and positive evidence that either Nouns or Verbs can have Prepositional Phrase complements. Note that the rules themselves need not be explicitly represented in this case.

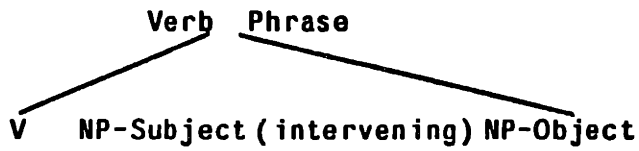
In particular, we have seen how *principles* such as X theory and the operator-operand structure implicit in the notion of "government" render many of the *rules* of grammar superfluous. On this account, what is acquired is not a specific rule system, but rather the knowledge to make a particular sequence of decisions. As we have seen, in certain cases this sequencing can itself shrink the amount of information it takes to represent the knowledge that must be acquired by a language learner.

Besides this advantage, there are engineering and descriptive reasons to aim at the elimination of re-write rules as a representation of knowledge of language. The descriptive reason is that languages other than English exhibit non-configurational dependencies. The term "non-configurational" here means that the relationships are *not* encoded via the structural patterns of phrases, but by some other means. A contrast with "configurational" encoding may help make this concept clearer. In English, the notion of "Subject" has been argued to be structurally defined, as the first NP under S. In contrast, in Japanese one cannot define "Subject" in this fashion, since the NP Subject need not be the first NP under S, or the second or third NP under S. Rather, the Subject is marked via a particle *wa* (sometimes *ga*) appended to the appropriate NP. Thus "Subject" in this case is non-configurationally defined. Re-write rules as conventionally defined cannot be easily used to represent non-configurational dependencies of this kind.

The engineering reason to eliminate re-write rules is to allow one to build a general-purpose parser and corresponding acquisition procedure. Moreover, this model fits naturally into a model of on-line semantic interpretation. Suppose that one's parser is built around a set of explicitly represented (context-free) re-write rules, or minimal extension of those rules, as is the case with most current parsing systems for natural languages. In languages where most or all relationships are configurationally defined, then this will seem to be the right representation to use -- hence the relative success of this approach for English. But in languages where structural relationships are impoverished then the work that re-write rules do is correspondingly small. In addition, there is evidence that certain languages possess "discontinuous constituents" -- that is, "phrases" that composed of non-adjacent elements at the surface. One way to picture this idea is that the "complete" phrase is reconstructed at an abstract level of representation, via an intermediary function Φ:



For example, Arabic has been claimed to have discontinuous Verb Phrases:



Even English exhibits this phenomenon. We have assumed that the Complementizer node is the Head of S. This seems partially correct since it permits *for* to govern *John* in sentences such as, *It would be foolish for John to go.* But the Head of S should be INFL if X theory is strictly followed -- recall that INFL is the Head of S; hence, by X theory, it should project to S. One way out of this puzzle is to claim that S is in fact a discontinuous constituent, assembled out of both COMP and INFL elements. There is some evidence that this is correct for languages such as Dutch; see Stowell [1981] for details.

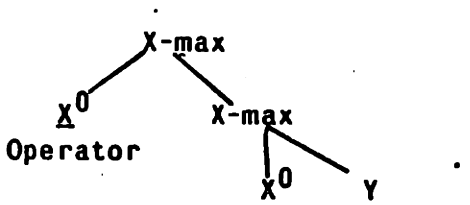
Plainly, it is the mapping Φ that does the work of re-assembling the complete phrase, and thus it is the specification of Φ that is crucial to this approach. Unless Φ is restricted, then the introduction of discontinuous elements weakens one's theory enormously, since a wide variety of non-natural languages could be described; see Chapter 4 for a partial formalization and a discussion of results of this kind.

In this section we will present an alternative conception of parsing that could be used to replace an explicitly-represented re-write system. This model will be able to handle languages that have relatively free word order, compared to English. Completely "free order" languages, such as Warlpiri, will still not be analyzable; this is a direction for future research. The resulting parser will still have access to acquired knowledge of syntax of course -- for example, Head-Complement order, and subcategorization information -- but it will not represent or make explicit reference to rules such as VP⇒V NP (PP). As an application of this approach, we will consider the analysis of simple Japanese sentences.

The key idea behind the alternative approach is this. Instead of regarding re-write rules as the primitive objects of our rule system, let us define the Operator-Operand relationship as primitive. If an Operator X takes an Operand Y as an argument, we will say that X governs Y. Crucially, government is defined so that it cannot "pass through" a maximal projection of X. Again, this notion is drawn from a natural *computational* restriction on the Operator-Operand relationship, namely that

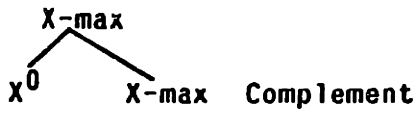
semantic interpretation is done "on-line," as every maximal projection is built. A maximal projection, like a Noun Phrase, forms a complete semantical unit. Assuming now a model-theoretic semantics, semantic interpretation rules take as input maximal projections and return appropriate expressions corresponding to the denotations of those maximal projections. For example, an appropriate object corresponding to an NP could be a pointer to a database representation of the object denoted by that NP. Crucially, the internal *syntactic* structure of the NP would no longer available, but just the single opaque object returned by the interpretation routine. Note that the analogy here to the on-line interpretation of programming languages is quite close. In the case of programming languages, input strings are parsed and replaced with their "meanings," generally, lines of intermediate code, abstract syntax expressions, pseudo-machine code, or the like. After translation, the original syntactic details are lost.

Now observe that if translation occurs maximal projection by maximal projection with opaque elements replacing internally-structured phrases as we go, then operator-operand control of Y by the X^0 underlined below cannot occur:

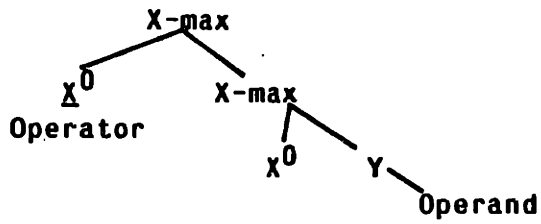


The reason is that the maximal projection that forms the Complement of $\underline{X}^0$ will be interpreted on-line as a single, opaque element. Therefore, any of its subconstituents will be inaccessible to further syntactic manipulation. The argument Y will be lost. In this way, maximal projections "block" the operator-operand relationship, an effect that is derived from an assumption about how semantic interpretation works. Thus the maximal projections themselves are parasitic on the notion of the Operator-Operand relationship, and serve no purpose aside from this.

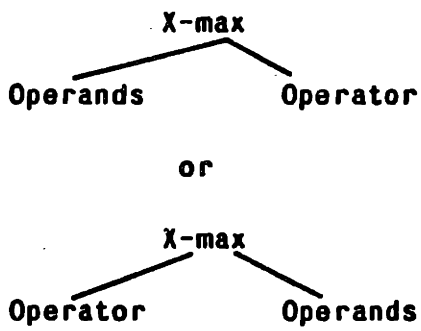
Observe that so far we have said nothing about how the Operator-Operand relationship might be established. In some languages, such as English, it is configurationally defined, apparently by a requirement of strict adjacency on the Operator-Operand relationship. Assume that this is the case, and assume further that semantic interpretation proceeds as suggested above. Then, the adjacency restriction means that in English maximal projections will have their characteristic branching structure:



But there is nothing to prevent the Operator-Operand relationship from being established by other means. For example, as noted above, Japanese uses particle markers as cues to connect the right operands to their operators. In any case, intervening maximal projections would not be permitted, e.g., structures such as,



That is, if we assume on-line interpretation, then Japanese should have a relatively "flat" constituent structure:



In fact, it has been argued that this is the case; see Farmer [1980].

Formally, let us say that if operand Y is to be evaluated with operator X then Y is associated with X, or that Y is bound to X. The notion of binding is again taken from the theory of programming languages, specifically that of operator-precedence parsing, and is meant to establish the required order for "semantic" interpretation, in this case the replacement of input strings by, e.g., object code. Consider by way of example an arithmetic expression such as, 2*3+5. As is well-known, an implicit parenthesization is established if we say that the multiplication operator, *, has a greater binding power than the addition operator, +. The resulting expression would then be parsed as, (2*3)+5.

Conversely, if + were made to have a greater associative "pull" than *, then the expression 2*3+5 would be analyzed as, 2*(3+5). Note that in this last case that 2 is combined with the *result* of applying + to the arguments 3 and 5, and not to 3 or 5 itself. Thus the internal details of the phrase 3+5 are inaccessible at this point, just as was suggested in the case of natural languages. Moreover, observe that the parenthesization need not be explicitly realized. It is the operator-operand relationships, not the parenthenses, that are "real," since it is a representation of these relationships that is acquired, stored, and accessed.

Conventionally, if an operator binds its arguments more tightly than another operator, we say that that operator *takes precedence* over the other operator; we want that operator and its arguments to be assembled into the same local phrase (conventionally called a "handle") before the second operator is ever applied.[71] Again by convention we notate the "takes precedence" relation by the symbol ·> The aim is to use the ·> relation to delimit the right end of a sentential form -- the tail edge of a phrase. (Clearly, any terminal element must take precedence over the end-of-sentence marker. The acquisition procedure uses this relation as one way to "fix" the end of a phrase.)

What about the left end of a phrase? By convention, we use the symbol <· to denote the left end of a handle. Finally, the symbol $\doteq$ is used to denote elements that appear as sisters of the same handle, intuitively, elements of the same operator-operand phrase.

In short then, if an operator θ_1 binds more tightly than an operator θ_2, then $a\ \theta_1\ b\ \theta_2\ c$ should be parsed as $(a\ \theta_1\ b)\theta_2\ c$. There is a large body of theoretical and practical results showing how to compute the relations ·>, <·, and $\doteq$, given context-free grammars in one or another restricted form.[72] However, the system presented here has no explicitly given grammar in the form of re-write rules. Still, it is possible to define precedence relations. Aho and Ullman [1977] suggest two heuristics:

---

71. A *handle* with respect to a right-most derivation may be defined as follows: "a handle of a right-sentential form γ is a production A-->B and a position of γ where the string B may be found and replaced by A to produce the previous right-sentential form in a right-most derivation of γ."
Aho and Ullman [1977, page 151]

Recall that the parser used in this work follows a right-most derivation for the most part, except for cases where an entire NP is parsed and placed into the second buffer cell. But even in this case, the order of attachment to elements in the active node stack follows a right-most derivation sequence.

72. For example, one might demand that all re-write rules of the grammar be of the form, A OP B, where OP is a terminal element that serves as an operator, A,B non-terminals. A context-free grammar in this form is called an *operator precedence* grammar. Interestingly, all context-free grammars have strongly equivalent operator-precedence grammars, so this restriction may not be a limitation in practice. See Gray [1969] for an overview.

> 1. If operator θ_1 has higher precedence than operator θ_2, make $\theta_1 \cdot\!> \theta_2$ and $\theta_2 <\!\cdot \theta_1$. For example, if * has higher precedence than +, make * $\cdot\!>$ + and + $<\!\cdot$ *. These relations ensure that, in an expression of the form E+E*E, the central E*E is the handle that will be reduced first.
>
> 2. If θ_1 and θ_2 are operators of equal precedence, then make $\theta_1 \cdot\!> \theta$ and $\theta_1 \cdot\!> \theta_2$ if the operators are left-associative, or make $\theta_1 <\!\cdot \theta_2$ and $\theta_2 <\!\cdot \theta_1$ if they are right-associative. [1977, page 161]

Observe the determination of whether one operator has higher precedence than another, or whether operators are left- or right-associative need not require an explicit grammar.[73]

Once precedence relations have been established, then parsing is a simple matter, barring any conflicts. A complete phrasal unit is set off by the symbols $<\!\cdot$ on the left, and $\cdot\!>$ on the right; $\doteq$ must hold between all the interior elements of a phrasal unit. In the natural language case, a phrasal unit will be an X category of some sort. To continue the parse, we replace the elements bracketed by $<\!\cdot$ and $\cdot\!>$ by the appropriate X$\cdot$ category -- as determined by the identity of the Head -- and recalculate precedence relations that may have been changed as a result.

Let us see how these ideas work in the context of a specific language, namely Japanese. The model presented here is highly speculative, of course; it is currently under implementation.[74] Still, the model illustrates the basic point of the discussion: explicit re-write rules need not be required as a representation of linguistic knowledge, even for parsing.

First, some basic facts about Japanese must be reviewed, drawn from the work of Farmer [1980] and Kuno [1973]. Japanese is, first of all, a Complement-Head (Postpositional) language: every sentence is terminated with a Verb, sometimes a question marker (*ka*). Noun Phrases are terminated by a thematic marker, *wa* (theme) or *ga* (Subject), *ni* (Indirect Object), *o* (Direct Object), among others. Relative clauses precede their head nouns, without relative pronouns or conjunctions. So for example:

---

73. We also need to defined precedence relations for adjacent elements that are *not* operators -- if the underlying system is not in operator-precedence form.

74. A preliminary implementation of this system has been completed by Alperin [1982].

```
Mary-ga        kaku   tegami-wa       omosiroi.
Mary-Subject   writes letters-Theme   are interesting.

[[[[Mary] writes] letters] are interesting]

"Letters that Mary writes are interesting."
```

Evidently, Japanese is basically left-associative. Observe that each phrase ends in an operator, if the class of operators = a Head in the sense of $\bar{X}$ theory, such as V, N, or particle.

We can use the heuristics suggested above to define precedence relations. For example, in the sentence above, V ends an X-phrase, so V ·> N. An X-phrase preceding a V is usually part of the operand structure of that V, however, so we have X-phrase [+N] $\doteq$ V. Similarly, an S-type X-phrase $\doteq$ an X-phrase headed by an N operator. A Noun particle (*wa, ga, ni, o*) takes precedence over a following N or V, i.e, Particle ·> {V, N}, but a particle and its preceding Noun are part of the same Operator-Operand structure, so N $\doteq$ Particle (type N). The analysis of the sentence above would then go as follows:

```
#   Mary   ga   kaku   tegami   wa   omosiroi   #

# <· N   ≐ ga·>  V

# <· X-max  ≐   V    ·>  N

# <· X-max  <·  N  ≐  wa  ·>  V

# <· X-max (S)  ≐ X-max (N)  ≐   V  ·>  #

#  <·  X-max  ·>  #
```

Or, to take a simple declarative sentence example,

```
John   wa    Mary     ni   hon     o          yatta.

John  theme  Mary  dative  book  direct object  gave

"John gave a book to Mary"

#  N  wa  N  ni  N  o  V  #

#  <·  N  ≐  wa  ·>  N

#  <·  X-max (N)  <·  N  ≐  ni  ·>  N

#  <·  X-max  ≐  X-max  <·  N  ≐  o  ·>  V

#  <·  X-max  ≐  X-max  ≐  X-max  ≐  V

#  <·  X-max (S)  ·>  #
```

We simply note here that the structure obtained is in line with Farmer's [1980] proposal about the structure of Japanese. Thus the scheme developed above can be considered as an instantiation of Farmer's approach as a parsing model.

This simple procedure requires substantial modification to handle more complicated cases. In particular, consider the sentence,

```
John  wa   Mary  ga       kaita   hon    o    yonda

John theme Mary Subject  wrote   book Object read

"John read the book that Mary wrote"
```

Apparently, some kind of look-ahead mechanism is required in order to distinguish this example from, e.g, *John wa hon o Mary ni yatta.* An extended precedence mechanism of the kind required is discussed by Alperin [1982]. In effect, it uses the look-ahead buffer of Marcus [1980] to calculate precedence relations as required.

Finally, the account of operator-operand interpretation presented here can even be used to explain the linguistically-motivated constraint of constituent command or c-command -- given our assumptions about on-line interpretation.[75] Briefly, a constituent α c-commands a constituent β iff the first

---

75. The following material is drawn from Berwick and Wexler [1982]. I am indebted to K. Wexler for discussions on this topic.

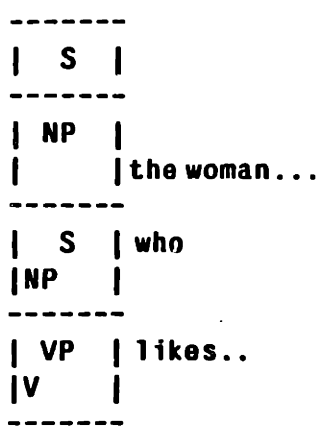
branching category that dominates α dominates β. The crucial linguistic observation is that an operator -- e.g., a quantifier expression -- must c-command the operand (variable) with which it is co-indexed. Indexation is conventionally notated by subscripting. So for instance, in the (a) sentences below the operators c-command their variables with co-indexing as shown, and are grammatical; in contrast the (b) sentences violate the c-command constraint on co-indexing and are ungrammatical as predicted.

(a) Every$_i$ man thinks that he$_i$ is something special.
(b) *The woman who likes every$_i$ man thinks that he$_i$ is special.

Now, this constraint has a very natural interpretation in the Marcus parsing model, and its modification used here. Recall that a maximal projection is built up when it is on the active node stack. When it is completed, it is dropped into the first position of the input buffer, only to be attached as the daughter of yet another constituent. At the point where a node is completely built and dropped into the input buffer then, its argument structure has been entirely assembled.[76] Therefore, this is a plausible point at which to interpret the maximal projection. (This arrangement is exactly the "pipelined" interpretive process that Marcus [1980] implemented.) What is left in the input buffer to be attached to some mother node is a single, opaque object -- an NP, a VP, a PP, or an S.[77] In contrast, any node in the active node stack, including the mother node to which the just-completed node in the buffer will be attached, is still accessible to further syntactic manipulation. This is as it should be, since we indeed still have to attach constituents to any node in the active node stack; this is why the nodes in the stack are called "active." For instance, given the sentence, "The woman who likes every man thinks...," at the point where "every man" is being analyzed, the active node stack will look like this:

---

76. We ignore here the problem of Preposition stranding, e.g., *What did you decide on?* This is an exception phenomenon, not observed in many languages. See Weinberg and Hornstein [1981] for an account that would retain the "constituent completeness" view suggested above, by incorporating the Preposition into the main Verb as a single word.
77. This constraint also does some of the work of the Freezing Principle advanced by Wexler and Culicover [1980] in order to guarantee the learnability of a family of transformational grammars.



Each of these nodes is active because it has not yet been completely built. The NP "the woman..." has not yet had its complement relative clause attached. The relative clause "who likes..." is unfinished because its Complement is not yet attached. Finally, the VP "likes..." is incomplete because its Object NP is not yet attached either. Because these nodes are not yet completely built, they are accessible to further grammar rule actions (barring additional constraints). As a result, any nodes attached to them are also accessible. Recall that these must be interpreted (opaque) nodes that have previously been dropped into the input buffer, only to be attached -- NP's, PP's, VP's, AP's, and S's. Therefore, it will not be possible to access any subconstituents of these nodes, by the definition of syntactic opacity. But this means that if an NP is to be a potential antecedent of a pronoun, then it must be visible in the active node stack, either a member of the stack or one node removed from a member of the stack. But the nodes so defined to be visible meet the c-command restriction. We conclude from this analysis that the c-command constraint is naturally obeyed, given certain assumptions about how semantic interpretation and parsing work. Further, in Berwick and Wexler [1982] it is demonstrated that this restriction helps speed up parsing, in that the number of primitive computational steps required for co-indexing is reduced in some cases by an exponential factor.[78] To sum up, we have obtained a possible two-fold *functional* explanation for the existence of this constraint: it comports with on-line semantic interpretation, and reduces the time required for co-indexing.

78. More precisely, if the parse tree is symmetric, then in the best case the number of steps to find the right antecedent is reduced from n to log (n), where n = the length of the input string up to the point where the pronoun is encountered.

3.5 Summary and Main Points

o A theory of acquisition complexity based on the notion of program size as applied to linguistic systems has been advanced.

o Non-instantaneous and instantaneous models of acquisition were compared. It was found that in some cases, even in the acquisition of finite rule systems (e.g., segmental systems), non-instantaneous models are advantageous for acquisition. By positing acquisition as taking place over time one can provide a better account of the existence and non-existence of segmental systems. Certain segmental systems are "inaccessible" because they are not reachable via natural developmental pathways.

o The non-instantaneous model can be envisioned as an ordered sequence of decision points moving through a lattice of possible systems. In this respect, it resembles concept acquisition models proposed in Artificial Intelligence research under the heading of "version spaces."

o The non-instantaneous model of acquisition also predicts the rough surface appearance of the development of segmental systems. This suggests that observed developmental stages may simply be the reflection of the underlying "choice points" of the non-instantaneous model.

o A general learning principle, the Subset Principle, was proposed. The Subset Principle accounts for a wide range of more specific learning principles that have appeared in the linguistics literature as constraints on evaluation procedures, e.g., that arguments are assumed by default to be obligatory, or that it is assumed by default that there is at most one surface structure for every deep structure.

o The Lparsifal acquisition procedure was formalized as a generalization of "k-tail" finite automata induction, now extended to use the Marcus parser as its definition of state equivalence. The Aux system was analyzed using this approach, and the various approaches to Aux compared from the standpoint of this formal model. All of the approaches -- phrase structure, node admissability conditions, complex V analysis -- were judged to be on a par, at least with respect to acquistion complexity.

o A way to replace an explicit re-write rule system as a representation of knowledge as accessed for parsing was proposed. A core set of principles, the $\bar{X}$ theory, Operator-Operand relations, and the Projection Principle, were advanced as a replacement for an explicit system of re-write rules. A simple parser based on these notions was described and applied to the case of Japanese.

4. Chapter Four

4.1 Modularity, Expressive Power, and Succinctness

4.1.1 Introduction: Modularity and Acquisition Complexity

Chapter Three sketched out a theory of *program size* or Kolmogorov complexity as one formalization of the notion of evaluation measure in linguistic theory. As discussed in that chapter, there is another complexity measure that has been more widely applied in the mathematical study of linguistic theories: *resource* complexity. This is the maximum amount of computational resources required to recognize whether a sentence is or is not in a given language, typically as measured in terms of the time or tape used by a Turing machine that recognizes the language. As was also pointed out, it has been (largely tacitly) assumed that resource complexity measures can be a useful gauge for linguistic theories because of a particular functional argument: if the theory of linguistic competence is to serve as the basis for actual language use, then the complexity of language recognition in the cognitive sense -- how people actually recognize, and parse sentences of their language -- might be presumed to be grounded upon the recognition algorithms mathematically possible for that language as specified by the theory of grammar. If one assumes in addition that one of the "design criteria" for language in the selectional sense is ease of recognition, it might then follow -- given some additional assumptions about how directly the theory of grammar is embedded in a model of language use -- that resource complexity would serve as a criterion for evaluating linguistic theories. In this view, theories of grammar that specify languages for which there are no efficient recognition algorithms would be less highly valued than theories that specified only efficiently analyzable languages.

In fact, this implicit argument seems to be quite widely held in the linguistic, and, not surprisingly, the computational linguistic literature. For example, this may be part of the motivation behind Bresnan's view that the theory of transformational grammar is not "psychologically real"; see Bresnan [1978] and Berwick and Weinberg [1982b] for discussion of Bresnan's position. The following quote from Marcus [1980] is a typical statement of the assumption that the linguistic system may have been designed according to the functional criterion of efficient parsability.

> But there is another fact about language behavior that is only slighly less marvelous: that language works at all. It is far from apparent how the mind, given only the speech waveform, or even a string of written words, can reconstruct linguistic structure in something like real time. From this, it seems reasonable to assume that language must be constrained in ways which make it amenable to efficient generation and recognition. This assumption, in turn, leads to the metatheoretic criterion that the more a given linguistic theory constrains the computational requirements for generating and parsing from the grammars it allows, the better the theory is.)

However, there is a problem with exclusive reliance on resource complexity measures. Suppose that

natural languages have been designed primarily so that they are easily *acquired.* If so, then it could be that after imposing acquisition restrictions one is left with a theory that specifies grammars that generate only efficiently analyzable languages. No additional constraint would then be imposed by the additional demand of efficient parsability.

Instead of developing a theory that ranks grammars according to resource complexity metrics then, one might aim at theories that use a metric of acquisition complexity. That has been the central concern of this thesis. If it makes sense to say that a grammar is "unrealistic" because it generates a language that cannot be efficiently analyzed, then it is likewise acceptable to say that a grammar is "unrealistic" if it cannot be acquired. It is important then to investigate whether the functional demand of acquisition is so restrictive as to subsume any demand that might be forthcoming from the constraint of efficient parsability. In the remainder of this chapter we shall investigate this matter in some detail.

First of all, it should be apparent that the supposed functional demands of learnability and parsability could be inter-related in complicated ways. In fact, there is a tradeoff between resource complexity and acquisition complexity that is familiar in the literature of automata theory but not so well known to linguistic theorists. Briefly, it can be shown that increases in generative capacity are generally coupled with decreases in succinctness, or the size of the representation required to specify a given set. More powerful formalisms "shrink" the size of descriptions for grammars (in the program complexity or information sense) but demand more in the way of computational resources for the languages they generate. But what of the tradeoff between these two demands? Below we will show that in several linguistically relevant cases more is gained in terms of simplified acquisition than is lost in terms of efficient recognition.

This automata-theoretic perspective will allow us to formalize another intuition that is common in the linguistic literature -- the role and power of modularity in linguistic theories (or scientific theories more generally). The term "modularity" here is used in two senses. First, a theory of grammar can specify a number of different linguistic levels, such as a level of syntactic representation, phonological representation, etc. Each level is a distinct language, that is, a set of expressions with its own vocabulary, principles, etc. Second, even *within* a single representational level, it may be possible to decompose a grammar into s set of interacting constraints. The first kind of modularity will be called *inter-level modularity*; the second, *intra-level modularity.*

An algebra for the first kind of modularity was originally discussed in Chomsky's *Logical Structure of Linguistic Theory* (LSLT). In this chapter it will be shown precisely how additional representational levels add to the power of one's theory (in the generative sense), and how much one can expect to "shrink" the size of descriptions of grammatical phenomena given one or more additional representational levels. The analysis leads directly to a formal, information-theoretic definitions of the

terms *stipulation* and *generalization*; these are presented in Section 4.2. Roughly, a stipulation is a "program" that achieves no compaction in the description of a set beyond a simple list of the elements of that set; in contrast, a generalization is a more succinct representation of a set than a list of the members of the set itself.

These simple formal tools will then be used to take a fresh look at the original arguments for additional levels of representation (the level of phrase structure and the level of transformations) that first appeared in LSLT. Section 4.3 demonstrates that the addition of a level of phrase structure (beyond simple string adjacency co-occurrence restrictions) allows for generalizations that a finite-state system can express only by stipulation. In other words, even if the set of surface strings of natural language formed a regular set, it could still be the case that the level of phrase structure would capture generalizations about attested natural language constructions that were not expressible in a finite state formalism except via stipulation. What is gained by incorporating an additional level of representation is explanatory power. Section 4.3 also discusses the *range* of possible compression possible by using a push-down automata representation of regular languages, as well as the possible advantage of using non-deterministic representations of linguistic knowledge. In brief, the non-terminals of a finite state grammar act as state variables; by introducing a notion of a *dependency relationship* that goes beyond simple string adjacency -- whether this notion is expressed by trees, hierarchical government, or co-indexing -- we can dispense with a large number of these variables. If we identify *acquisition complexity* with the amount of information that must be specified in order to fix a grammar, a specification that is bounded from below by the number of non-terminals in a grammar, then the succinctness gains of a phrase structure reperesentation translates into a significant lessening of the acquisition burden.[1]

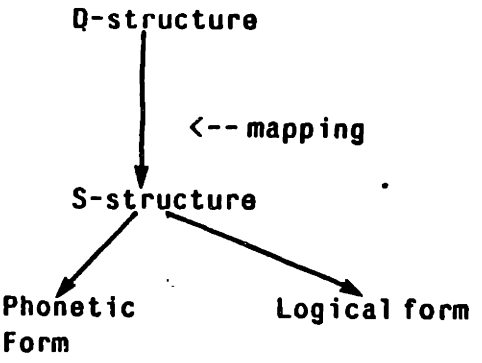
Section 4.4 extends the succinctness analysis to the case the extended phrase structure machinery proposed by Gazdar [1981]. It shows that the Gazdar phrase structure theory has the same problems in accounting for the effects of moved constituents as finite state machines have in accounting for hierarchical phrase structure. In effect, Gazdar attempts to "simulate" a transformational level by

---

1. Interestingly enough, the recent development of phonological theory has been marked by the same effort to adopt more powerful kinds of dependency relationships in order to eliminate the need for state-variables in grammars. As long as it was assumed that phonological systems had to rely on string adjacency as the sole means of stating generalizations about phonological rules, then it also seemed necessary to use some sort of *variables* in order to "remember" different state possibilities. (Example:) But as soon as devices more powerful than string adjacency are admitted -- be they metrical trees, or a tier system, or whatever -- then the need for variables is diminished, even eliminated. In retrospect, and considering the formal analysis to be presented below, this should have come as no surprise. Similarly, the non-adjacency based theory of phonology seems determined to repeat the scientific history of transformational theory: just as early transformational theory had an extremely rich repertoire of operations, including quantification conditions triggering conditions, arbitrary Boolean conditions on analyzbility, and the ability to construct quite complex transformations out of several elementary transformations, only to advance to a quite restricted system where none of these options is possible, so too the non-adjacency based theory of phonology appears to be moving from a relatively unrestricted theory of tree-like operations to a more restricted account. See Vergnaud and Halle [forthcoming] for additional discussion.

encoding (stipulating) derivational history. Thus extended, phrase structure machinery can perfectly well *describe* the same set of surface strings (and even the same set of trees) as a modern theory of transformational grammar; however, in some cases the description demands that one add a series of stipulations to the theory (in a formal sense of stipulation). The resulting grammar has no more explanatory power than a list, again in a formal sense of explanatory power. Thus one can show that modularity in the sense of distinct "levels of representation" serves as a powerful device to "shrink" descriptions of linguistic knowledge, hence acquisition complexity.

Sections 4.5 and 4.6 sketch automata theoretic models of inter- and intra-level *modularity*. *Inter*-level modularity obtains when a theory can be described as a series of mappings between one or more distinct representational levels -- as for example in the theory of transformational grammar, where one assumes separate levels of deep structure and s-structure, along with a mapping connecting the two levels. In addition, current theories add separate representational levels of phonetic form (PF) and logical form (LF), also connected to s-structure by mappings:



Note that this approach assumes that representational levels are simply sets of *strings* rather than, say, trees. It thus adopts the model of representational levels of LSLT or Lasnik and Kupin [1977].

By suitably characterizing the mapping between representational levels as a restricted kind of transduction, in this case a generalized sequential machine (gsm) or push-down transduction, one can place this model in an automata-theoretic context. In particular, one can show that a push-down transduction characterizes the sense in which an additional level of representation, e.g., a level of phrase structure or a level of transformations, adds extra power, hence potential succinctness, to a theory.

Intra-level modularity obtains when a theory contains several independent principles all operating at a single representational level. For example, in some formulations of the Government-Binding theory [Chomsky 1981] the sub-theories of government, bounding, and indexing can be taken to operate at the level of s-structure. As we shall see, it is more advantageous from the standpoint of of acquisition

complexity to incorporate them as independent principles than to encode them, as in a Gazdar-type system, into an expanded context-free rule system. In general it can be shown that the encoding of independent principles into a single set of re-write rules results in a multiplicative expansion in the number of rules required, and hence a corresponding expansion in acquisition complexity. Indeed, if anything like the Government-Binding decomposition of linguistic knowledge is correct, then the entire enterprise of using re-write rules as the cornerstone of one's linguistic theory must be viewed with some suspicion. Context-free re-write rules are a notational machinery that may be used to *describe* certain co-occurrence relationships. As we shall see, in certain co-occurrence constructions, notably those falling under the general heading of co-indexing, as in the binding of a trace by displaced constituent in a non-argument position, or a quantifier and the variable it binds, it will be shown formally that a phrase structure account can do no more that simply describe, not explain, such phenomena.

4.2 The tradeoff between acquisition and recognition complexity

Suppose that we increase the *generative power* of our linguistic theory, in the automata-theoretic sense of generative power. That is, suppose that we enlarge the class of languages that a theory can generate. For example, if we move to a formalism that is able to generate all the context free languages, then we have more power at our disposal: we can generate all the regular languages, and all the context-free languages besides. Some researchers argue that the price paid in moving to a more powerful descriptive apparatus is unjustified; two reasons are usually given for this view, one an argument that explanatory power is decreased, the other than recognition complexity is increased. To paraphrase these arguments:

> (1) Since the class of languages that can be generated is broader, it follows, *ceteris paribus*, that explanatory power is decreased: we are less able to say why natural languages are the way they are. For example, since the class of context-free languages is a proper subset of the class of context-sensitive languages, then if it is also the case that context-free languages are descriptively adequate to accomodate the phenomena found in natural languages, we should opt for the more restrictive theory -- i.e., a theory that can generate only context-free languages.
>
> (2) Similarly, it is well known that classes of languages that are broader in the automata theoretic sense have members that, in general, are more difficult to recognize in terms of time- and tape- bounded automata. (Indeed, an r.e. language need not even have an algorithmic recognition procedure for the non-sentences of the language.) We shall consider these results in depth shortly. Since people apparently recognize sentences quickly, it follows, so this line of argument goes, that if one demands that the theory of linguistic competence provide some sort of direct account of

> linguistic performance, then one cannot opt for a theory that admits natural languages whose members cannot be quickly recognized algorithmically.

Both these arguments are inconclusive, however. The first neglects the possible effects of the *ceteris paribus* assumption: while it is true that an increase in the power of the formal apparatus allowed broadens the general class of languages describable by a theory, it is also true that this increased power can permit a more compact statement of the description of particular languages. As we shall see, this gain in succinctness can more than make up for the loss in explanatory power assumed under (1) above.

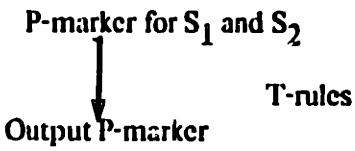
The second argument, based on recognizability, is also flawed, for the most part because it fails to recognize a possible gap between the theory of grammatical competence and theory of performance. For example, people cannot analyze arbitrarily deep center-embedded constructions, but it does not follow that context-free grammars are thereby ruled out as characterizations of linguistic competence.

To take an informal and familiar example of a gain in explanatory power that comes from adopting a more powerful theory of grammar, consider the argument (originally advanced in *The Logical Structure of Linguistic Theory*) for positing an additional "linguistic level" of representation beyond that of phrase structure. The difficulty with phrase structure, as Chomsky argued in *Syntactic Structures* is its apparent inability to capture certain generalizations about language, for example, conjunction:

> We can simplify the description of conjunction if we try to set up constituents in such a way that the following rule will hold:
>
> (26) If S_1 and S_2 are grammatical sentences, and S_1 differs from S_2 only in that X appears in S_1 where Y appears in S_2 (i.e., S_1 = ..X..and S_2 = ..Y..),and X and Y are constituents of the same type in S_1 and S_2, respectively, then S_3 is a sentence, where S_3 is the result of replacing X by X+*and*+Y in S_1...
>
> Even though additional qualification is necessary here, the grammar is enormously simplified if we set up constituents in such a way that (26) holds even approximately. That is, it is easier to state the distribution of "and" by means of qualifications on this rule than to do so directly without such a rule. But now we face the following difficulty: we cannot incorporate the rule (26) or anything like it in a grammar...of phrase structure, because of certain fundamental limitations on such grammars. The essential property of rule (26) is that in order to apply it to sentences S_1 and S_2 to form the new sentence S_3 we must know not only the actual form of S_1 and S_2 but also their constituent structure--we must know not only the final shape of these sentences, but also their "history of derivation". But each rule X⇒Y of the [phrase structure reb] grammar applies or fails to apply to a given string by virtue of the actual substance of this string. The question of how this string gradually assumed this form is irrelevant.

[1957, pages 36-37]

By adding another level of representation -- that of transformations -- we can easily refer to the history of a derivation, as recorded in its P-marker. Since T-rules operate on P-markers, we can make an output P-marker contingent upon derivational history:



Below we shall make precise the sense in which this increases generative power, by showing exactly *how* the addition of levels of representation increases descriptive power in the automata theoretic sense.

It is important not to misinterpret the argument that is being made here. It is technically possible -- as observed by Harman [1963] and revived by Gazdar [1981] -- to use a bookkeeping mechanism such as subscripting the non-terminals of a context-free (phrase structure) grammar to encode a *finite amount* of derivational history as part of the phrase structure grammar itself. However, as we shall make precise in discussion below, this extension of the usual notion of phrase structure is in effect simply *simulating* the computational power of adding an additional level of representation -- that of transformations We shall demonstrate that this extension is precisely analogous the more familiar case of truncating a push-down automaton's stack at some finite depth and then claiming that the resulting device is "really" a finite-state machine. It is true that such restricted machines can accept only regular languages. But such a device is not a finite-state machine; rather, it is a machine whose input-output behavior can be *simulated* by some *other* device that is a finite-state machine. Moreover, for certain constructions actually found in natural language (center-embedded constructions) this simulation can be shown to require an exponential growth in the number of states of the finite-state machine. It follows that the corresponding "grammar" of the simulating finite-state machine is exponentially larger than a grammar for the push-down automaton that it was simulating. If we take grammar size as a rough measure of the amount of information that must be "specified" in order to acquire the grammar, then the push-down formalism is exponentially more compact, hence requires an exponentially smaller amount of information for its acquisition.

There is a slighly different way to view this result. One can show that there can be no gain in explanatory power via the simulation of a push-down machine via a finite-state machine. This is so because in order to simulate a push down automaton via a finite-state mechanism, or simulate an additional level of transformations via phrase-structure subscripting, one must in essence add a table that takes up as much space as the set of all possible combinations of surface strings that could possibly

be observed.[2] This simulation is then but a stipulation that a finite portion of some set of strings "acts like" a push-down automaton, rather than a generalization that captures true push-down stack discipline. In other words, a simulation just stores all possible combinations of possibly observed surface arrangements of words (in the finite state simulation) or phrases (in the phrase structure simulation of T-markers).

More precisely, call the set of surface data to be accounted for (a set of sentences or phrases) S. Suppose for the sake of argument that S is of finite cardinality, and that the length of strings in S is n. There are only 2^n possible binary strings of length n, or, in general, k^n strings of length n for an alphabet of size k. Therefore, a machine with 2^n (k^n) states has the power to simply store *any* possible specification of S in a table of roughly the size of S. Given such a machine, then S has a trivial program description via a program that simply lists the members of S. The size of the program is about the size of S (denoted |S|), and as |S| approaches the full number of possible strings of length n, then the size of the program describing S must approach 2^n. If a description of S does no better than this trivial description, then we will call it a stipulation.

> **Definition.** Let $D_\Lambda(S)$ be a description of a finite set S wrt an acceptable programming language Λ, |S| the size of S. If the programming language is clear from context, we will drop the subscript Λ. $D_\Lambda(S)$ is a *stipulation* if the *size* of $D_\Lambda(S)$, denoted $|D_\Lambda(S)|$, $\geq$ |S|.
>
> If D is not a stipulation, then D is a *generalization.*

In other words, D is a stipulation if it takes as many bits to describe S as the size of S itself. In contrast, if D is a generalization, it is "shorter" than |S|. Now recall the definition of a random string from section 3:

> **Definition.** A finite string s is *random* if $K(s) \geq |s| - c$, where K(s)= the program size (Kolmogorov) complexity of s wrt an optimal programming language, c a positive integer fixed arbitrarily.

We see then that if $D_\Lambda(S)$ is a stipulation, we have, in effect, regarded the set S as if it were a random string; this is the formal sense in which a stipulation D "ignores" any possible generalizations about S. A simulation that uses an exponential number of states in n (2^n states) to describe strings of length n is a stipulation, because it does no better than simply storing the strings themselves.

For infinite sets of distinct strings we must modify this definition because there is no bound on the length of strings in such a set. The difficulty is akin to that in defining a size measure for Boolean circuits: we cannot say that a single circuit computes a function, since any single Boolean circuit can

---

2. In the case of an infinite table, one actually stores finite initial segments of the full table, a "family" of tables.

compute only the values of a function for fixed length arguments. Instead we say that a function is computed by a *family* of circuits, and make the construction of the family uniform. Similarly for infinite sets of strings:

> Let $\{S_n\}$ be a uniform family of sets, $S_n = \{\Sigma^n \cap S\}$ where Σ = the alphabet of the language; Σ^n = all possible strings over Σ of length n or less.
> (S_n is the just the finite portion of S consisting of strings of S that are n long or less.)
> A family of sets is *uniform* if there is some deterministic Turing machine that can compute the function $1^n \text{-->} S^*_n$ in space O(log c (S_n)), S^*_n a binary string coding the description of S_n. This definition follows that in Cook [1981].
>
> A description $D_{A,n}(S)$ is a stipulation if, for $\forall$ but a finite number of integers n, $|D_A(S_n)| \geq |S_n|$

To illustrate these definitions, consider the following language, the palindromes of length six over the alphabet {a, b}:

> S_1 = babbab, aabbaa, bbaabb, abaaba, baaaab, aaaaaa, abbbba, bbbbbb.

A machine (or table) with 2^n states can store any set of strings of length n. Thus, we can specify S_1 via a finite-state machine with size at most 2^6 (=64). However, with this size machine we could just as well have stored the structure of the following random set of strings of length six:

> S_2 = aaaaab, aabbaa, bababa, aabbbb, bbbbbb, abbbaa, aaaaaa, abbbba.

Thus, while the finite state simulation can *encode* the mirror-image structure of language S_1, it does so "blindly"; it could just as easily encode a non-mirror image structure. But of course, S_1 *does* have a mirror-image structure that can be specified by a push-down automaton with fewer than 2_6 states.

> We build the corresponding pda in two steps; first we build a pda that accepts *all* even length palindromes; then we add a "counter" that detects when the stack gets more than 3 deep, and rejects if it does. The basic pda is from Harrison [1978].
>
> stack symbols Γ = a,b, Z_0. (Z_0 marks the bottom of the stack.)
> initial state: (q_0, a, Z_0)
>
> transitions:
> $\delta(q_0,a, Z_0) = \{(q_1, Z_0,a)\}$
> $\delta(q_0,b, Z_0) = \{(q_1, Z_0,b)\}$ stack a or b initially.
> $\delta(q_1,a, a)$ = read and push a, with a on top of stack go to state 2

$\delta(q_1,a, b)$ = read and push a, with b on top of stack go to state 2
$\delta(q_1,b, a)$ = read and push b (do these have to be specified as well?)
$\delta(q_1,b, b)$ = read and push b
now repeat for one more push. then start popping. (4 more machine instructions)
$\delta(q_3, a, a)$= read and pop a; stay in state 3
$\delta(q_3, b, b)$= read and pop b; stay in state 3
$\delta(q_3, \Lambda, Z_0)$= accept

Total: 13 transitions.

(This account is deliberately vague about whether the size of a pushdown automaton is simply the number of transitions in the pda. Below we define the *size* of a push-down automaton more carefully as the total number of symbols in the pda specification, including the space needed to encode alphabet symbols.)

In general, the palindromes of length less than n over a fixed alphabet of size 2 will require a push-down automaton of size O(n) -- a fixed number of states to generate all even palindromes of arbitrary length, and n states to "count" up to n. But since the number of palindromes of length n for n even grows on the order of 2^n, the pda description amounts to a generalization about these languages, namely, their mirror-image structure.[3] In contrast, the finite state simulation of the pda for such languages demands $O(2^n)$ states, a stipulation. Below we extend this result to show that extensions of phrase structure designed to handle,-e.g., co-indexing via non-terminal subscripting are also stipulative, in just this sense.

The fact that a given set of strings may have a number of descriptions, with some better than others, simply restates the obvious fact that a single set can have many alternative representations, some "better" than others according to certain purposes. Any number of formalisms may prove to have sufficient power to simply *describe* a set of strings generatively. For example, suppose natural languages are only context-free. It then follows that given any natural language, there is *some* context-free grammar that describes that language. But it does not follow that this context-free grammar will necessarily give insight into why this particular language is built the way it is rather than some other way; equivalently, it may not provide succinct descriptions for the languages in question.

Turning now to the question of recognition complexity, it has sometimes been argued that admitting an additional level of representation is ill-advised because in general adding additional representation levels increases generative power, and, in turn, larger classes of languages demand additional

---

3. Given the even palindromes of length n-2, the palindromes of length n may be formed by choosing either *a* or *b* to add to each palindrome of length n-2. Therefore the number of palindromes of length n, $P_n = 2\,P_{n-2}$. This recurrence may be solved to obtain $P_n = 2^n$.

computational resources for their recognition. Let us briefly review the known results about recognition complexity that will prove relevant in the sequel.

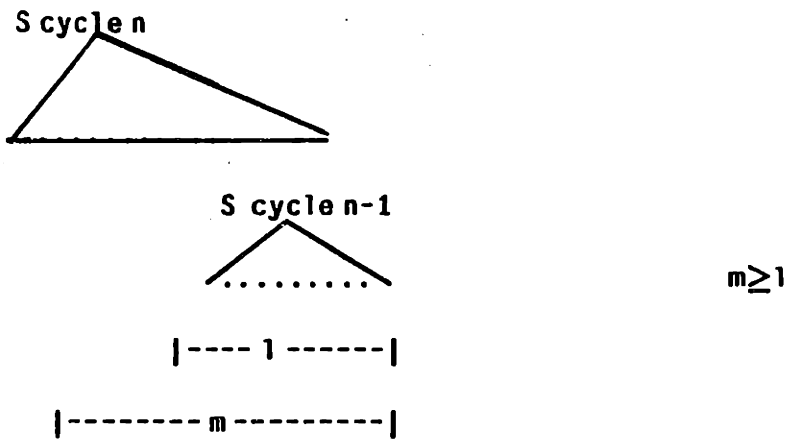
(1) Finite-state (regular) languages are recognizable in real-time on a deterministic Turing machine, that is, in time |n| where n=the length of input sentences in number of tokens. (Rosenberg [1967]). From the standpoint of complexity theory, what this means is that finite-state languages take no computational resources, aside from the cost of reading the input sentence itself. (This would seem to be a peculiar result from the standpoint of cognitive fidelity; it suggests, as we shall discuss below, that standard complexity measures are perhaps inappropriate as cognitively-relevant complexity criteria.)

(2) Context-free languages are recognizable (and parsable) in less than or equal to n^3 time; less than log-squared space; or in simultaneous polynomial time and log-squared space. (Recall that to *recognize* a language means to simply say whether or not the string presented is or is not in a specified language; the notion of *parsing* subsumes recognition since in addition we must provide a representation of the actual derivation of the sentence via the rules of some grammar for the language. Thus parsing is a grammar-dependent notion.) (Earley [1968])

(3) Context-sensitive languages are recognizable in space linear in the length of inputs (a linearly-bounded automaton) or in exponential (2^n) time or less.

(4) By comparison, the theory of Transformational Grammar (TG) as described in *Aspects of the Theory of Syntax* could generate any r.e. set Peters and Ritchie, [1972]. Hence, the theory allowed languages that do not even have algorithmic (recursive) recognition procedures -- for some sentences not in the language, the recognition procedure would loop forever.

(5) If one adds a restriction on an *Aspects*-style TG theory known as "terminal length nondecreasing constraint", then the languages generable by TGs turn out to be exactly the languages accepted in exponental (2^n) time. (Rounds [1975]; this is because one can show that such systems can be simulated by auxiliary push-down automata, that is, a push-down automaton that has an auxiliary, bounded tape. Note that such a device subsumes the power of a linearly bounded automaton, since it possesses a stack in addition to a linearly bounded tape. We shall discuss the reason for this extra power systematially below.) What this restriction comes to is that as the TG cycles from one S-level to the next higher S-level, the length of terminal material subsumed by the domain of the current cycle must always be at least as great as that the previous cycle:



where "...." denotes a string of terminals

(6) As for other recent theories of grammar, we observe that Berwick [1981] has shown that Lexical Functional Grammar recognition (Bresnan and Kaplan [1981]) is NP-complete, and hence probably demands non-polynomial time resources, just as in the case of TGs with the terminal length non-decreasing condition. (It is widely conjectured that NP-complete recognition problems can have no polynomial time algorithms; see Machtey and Young [1978] for discussion.) We see, then, that the gross categorization of resource complexity demands into polynomial or exponential classes probably does not distinguish between the theory of TG and LFG.

In general, one can conclude that adopting a more powerful (more expressive) formalism entails a concomitant increase in recognition complexity. That this should be so follows from the intimate relationship between classes of languages defined in terms of *structural complexity* (e.g.; depth of recursion, number of stacks) and classes of languages defined in terms of *resource complexity* as established by Ritchie [1963], Cobham [1964], McCreight and Meyer [1969], and others. Roughly, a function f is in a structually defined class C iff there is a program for f whose resource complexity is bounded by a function in that same class. (So for example, the elementary recursive functions are precisely those whose running times are bounded by elementary recursive functions.) Thus, by enlarging a class of languages, we enlarge possible candidate bounds on running time or space use.

There are two main fallacies with the argument that one should avoid enlarging generative power because of an associated cost in increased recognition complexity: (1) the possible cognitive irrelevance of the mathematical worst-case complexity results; (3) the possible off-setting gains in succinctness (sketched above) that impact on both acquisition and recognition complexity. Let us review the first briefly before turning to an extended discussion of the second.

The first two problems with the no-increase-in-generative power argument both have to do with the

difficulty of interpreting general mathematical results in a specific cognitive domain. The major points here have been covered extensively in Berwick and Weinberg [1982a]. To review these briefly here, observe first that the complexity measures associated with generative classes are *asymptotic* measures. That is, they hold only in the limit of the length of sentences. Thus, when one says that the sentences of a context-free language may be parsed in cubic time or less, one actually means that they can be parsed in time kn^3, where n = length of sentences input to the parsing procedure (in words), and where k = a constant that is dependent on the implementation details of the algorithm and the size of the grammar. For small sentences this function may well be dominated by the grammar size rather than the n^3 term. Hence, it could well be that for all sentence lengths that were of importance in the evolutionary design of language asymptotic functional terms were of little or no significance in determining parsing efficiency, but grammar succinctness was crucial. In such a situation more compact grammars would have a dual advantage, being both easily acquired and easily processed. In short, it is far from clear that a restriction of one's theory to some mathematically well-defined class of languages with certain recognition time properties has anything to do with cognitively relevant recognition time complexity or acquisition complexity. (As Berwick and Weinberg show, a more exact analysis of this matter hinges upon the algorithmic details embedded in the constant k as well.) The conclusion is that cognitive parsing complexity need not correspond to asymptotic resource complexity. The generative hierarchy need not correspond to the functional evaluation of complexity in the cognitively relevant sense.

Further, suppose it turned out to be the case that the theory of grammar for natural languages generated some languages that were, say, r.e. but not even context-sensitive. Nothing of consquence need follow. For example, we might simply note that people are unable to analyze some (non)sentences, or give up analyzing some sentences whose processing demands too much in the way of computational resources -- that is, exactly what is already known to be the case.

In the remainder of this chapter we shall focus on the third flaw in arguments against increasing expressive power, the possible effects of succinctness on acquisition and recognition complexity due to choosing a more powerful representational formalism.

4.3 Succinctness for finite-state and context-free grammars

As a case study in the power of additional representational levels, we shall take up in detail the case of the trade-off between push-down automata and finite-state automata for the representation of regular languages. This will provide the foundation for the more general study of phrase structure languages. We conclude with a series of formal results that specify in detail the additional descriptive power that accrues by adding additional levels of representation.

Consider then a finite regular set that is accepted by a nondeterministic (deterministic) push-down automaton with n states and s push-down symbols. This pda takes roughly n+s symbols to specify; let us say that it is of size n+s.[4] We first review a theorem established by Meyer and Fischer [1971].

Theorem: (Meyer and Fischer, 1971) *If a finite regular set is accepted by a nondeterministic (deterministic) pda with n states and s push-down symbols, then that set can be accepted by a nondeterministic (deterministic) finite state automaton with at most* $O(s^{(sn)^2})$ *states.*

Proof. We will show that if a push-down automaton accepts only a finite language L, then its stack can grow at most $(sn)^2$ deep. The result then follows immediately, because if the pda's stack can be at most $(sn)^2$ deep, a finite state automaton with $s^{(sn)^2}$ states can store all the possible stack configurations of the pda in its finite control, and simulate the pda.

To see why the pda's stack can be only $(sn)^2$ deep, consider number of possible <state--top of stack> pairs there can be before such a configuration is repeated. If there are n possible states and s possible stack symbols, then there are s x n possible state-stack pairs. But then, a machine accepting a finite language L can have at most (sn) x (sn) distinct state-stack configurations before repeating one. Suppose then that the pda's stack does grow deeper than $(sn)^2$. Then the machine must enter a duplicated configuration during the processing of some portion of a string in the language $w = w_1 w_2 \ldots w_n$. That is, the <state, stack> combination after processing w_i is the same as the (state, stack) pair after processing w_j, for some j>i. But by the assumption that the <state, stack> pair is the same after processing w_j as after processing w_j, it follows that the machine will also accept a string with the substring $w_i \ldots w_j$ arbitrarily "pumped". That is, it will also accept $w_1 \ldots (w_i w_{i+1} \ldots w_j)^* w_{j+1} \ldots w_n$. Therefore the machine will accept an infinite number of strings, contrary to assumption. ∎

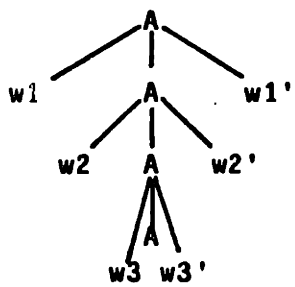
Thus, the potential gain in succinctness of a push-down automaton over a finite state automaton is at

---

4. There are two basic ways that we might measure the size of a pda. First of all, we could simply count the number of transitions in the pda -- as we did in the informal example for a finite palindrome langauge above. This would be the same as setting grammar size |G| to the sheer number of rules in G. This measure, however, does not take into account the length of the rules or the coding space required for alphabet symbols and state names -- proportional to the log of the number of such symbols, including stack symbols, and the number of states. If we let S be the total number of symbols required to write down the pda's transitions, this "normed" |G| could be calculated as S log ($|\Sigma|+|\Gamma|+|Q|$), where Σ = the input symbol alphabet; Γ = the push-down stack alphabet; and Q = the states of the transition. This adjusted |G| would thus measure the number of *bits* required to encode the pda. Clearly, the number of transitions must be a lower bound for S. Note also that for the exponential size changes considered above this logarithmic factor will not be relevant. In fact in this case, the product of states and symbols of the pda, ns, is just as "precise" a size measure as the more careful definition. To see this, let S be the total size of all transitions. As pointed out in the proof of the theorem above, there are at most $O((ns)^2)$ distinct transitions. Thus S is bounded above by $k\,(ns)^2$. But S is also bounded below by c (sqrt (ns)), because every state n and stack symbol s appear in at least one transition. Thus the product measure is at worst a small polynomial factor in error, compared to the more refined information theoretic measure.

most exponential.[5] Whether this potential gain is important for linguistic theory depends on the existence of any linguistically relevant examples that actually force a finite state automaton to use an exponential number of states.

In fact, one of the original examples demonstrating the inadequacy of finite state representations of linguistic competence provides an example of exactly this sort. Consider sentences generated with a finite number of self-embedding constructions (produced by rules of the form $A \Rightarrow \alpha A \beta$, α, β non-null). The resulting structures are typical of nested constructions in natural languages. To make the Meyer-Fischer theorem applicable, let us further restrict ourselves to the case of self-embedded sentences of only finite length. Then the co-ordination in self-embedded sentences can be represented abstractly as follows:



resulting string: w1 w2 w3 w3' w2' w1'

If we intersect the resulting language with a simple homomorphism that removes the primes on the w's then we get a finite palindrome language:

$\forall i,\ h(w_i) = w_i;\ h(w'_i) = w_i$

We can now show that if it were the case that the language consisting of self-embedded sentences of finite length n could be recognized by a finite state machine of size less than $O(2^n)$, then it would also follow that the palindrome languages of finite length n could be recognized by a finite state machine of size less than $O(2^n)$, a contradiction.[6] Suppose there were such a machine. Then we could use this machine to build a new machine of size less than $O(2^n)$ to recognize the finite palindromes nas follows: process the finite self-embedding language, and then apply the homomorphism *h* as defined

---

5. As Meyer and Fischer note, since the standard conversion from a push-down automaton to a context-free grammar causes at most a linear expansion in the size of the resulting description, this result holds for grammars as well.

6. That the palindrome languages require an exponential number of finite states for recognition can be established by a simple "adversary" argument: a machine must store on the order of $O(2^n)$ strings; otherwise, one could "fool" the machine into accepting a string when it should not.

above. The homomorphism is of size independent of the length of sentences. (If h is stored as a table, then there must be $O(|\Sigma|)$ entries in the table.) Thus the two-step machine will of size less than $O(2^n+m) = O(2^n)$. But the finite length palindrome languages of length n require finite state machines of size $O(2^n)$, a contradiction. ∎

Thus any finite state automaton for the finite, finitely self-embedded languages requires the specification of an exponentially larger number of states than a push-down automaton (hence context free grammar) for the same language. The stack of the push-down automaton saves having to encode all possible combinations of linear co-occurrence relationships. This demonstrates, formally, why it is that a context-free grammar is more compact than a finite-state representation for nested constructions.

Hierarchical co-ordination, then, actually realizes the potential exponential gain of pda representations over finite state representations. For what sort of languages is there no gain in moving from a finite state to a push-down representation? Not unexpectedly, there is no gain if the language has no hierarchical co-ordination, that is, each lexical item in a string is determined entirely by the item that precedes it. The structure of such a language consists of purely linear order; it is completely "flat" in terms of phrase structure. Such a language could exhibit no co-occurrence relationships other than string adjacency relationships; that is, it could not express co-occurrence constraints that cut across terminal items. Ginsburg and Lynch [1976] have shown that the following language exhibits these properties; it requires a finite state automaton of size O(n), yet any pda for the same language will also be of size O(n):

$$a_1^* a_2^* \ldots a_{2n}^*$$

Let us sketch out Ginsburg and Lynch's proof of why this is so. Clearly, a right-linear grammar for this language can be constructed of size O(n): the rules $S \Rightarrow A_1$, $A_1 \Rightarrow a_1 A_1$; $A_1 \Rightarrow A_2$; etc.; plus $A_{2k} \Rightarrow a_{2k} A_{2k}$; $A_{2k} \Rightarrow \epsilon$ suffice. But any context-free grammar for the same language will also be of size O(n). The idea of the proof is to show that any context-free grammar that generates this language must have at least n distinct non-terminals X_i. Consider a derivation tree for a long string in the language L that is sufficiently deep that it has a path with at least $v+2$ nodes, where $v =$ the number of distinct variables in the context-free grammar G for L. But then, some variable in the grammar, call it X_i, appears twice in the derivation of this sentence for L; i.e., $X_i \Rightarrow_G u_i X_i v_i a_i$. But then, there cannot be three distinct indices, $i < j < k$, such that $X_i = X_j = X_k$. This is because it would follow that some a_i could be generated out of order; for example, if X_k replaces X_j, with $j < k$, then the a_j associated with X_j is generated to the right of a_k. Since all i,j,k triples are distinct, by the pigeonhole principle it must be the case that out of the at least $2n$ indices X_i there are least n distinct indices. ∎

In sum, the following languages represent the extremes of possible succinctness gains of a pda over a finite state automaton for the same regular language:

(1) maximum succinctness gain (exponential): languages whose structure is determined entirely by nested, hierarchical constructions.

(2) minimum succinctness gain (non-): languages whose structure is determined entirely by linear word order.

(1) and (2) establish upper and lower bounds on the succinctness gains possible by using a pda representation instead of a finite state representation for a regular language. The succinctness gains for a pda (context-free grammar) representation over a finite state representation for all other regular languages must fall somewhere in between these two bounds, depending upon the mix of hierarchical and adjacency co-occurrence structure in the particular finite, regular language.

Having specified the possible gains due to a decrease in descriptive complexity, we must now address the question of a possible loss in terms of increased recognition complexity. What is the change in terms of recognition complexity by adopting a pda representation over a finite state formalism for the same language?

Recall first that any regular language can be recognized in real time, $|n|$.[7] Second, we note that the regular languages form a proper subset of the deterministic context-free languages; by a well-known result (see Hopcroft and Ullman [1979]), this means that there is a deterministic pda (corresponding to a parser for an LR(1) grammar) that accepts each regular language. (For example, the pda accepting palindromes of length six described above is deterministic.) But the running time of any LR(1) grammar is linear in the length of sentences, i.e., kn. By an elementary theorem of complexity theory, if we are allowed to recode the primitive instruction repertoire of our underlying computational model, we may speed up this running time so that k is as small as we want, and kn is as close to n as desired. As the preceding footnote observes, this ability to recode the instruction repertoire of the underlying Turing machine is also assumed in order to obtain a recognition time of exactly $|n|$ for the languages recognizable in real-time. We conclude that, mathematically at least, there need be no *perceptible* loss in recognition time complexity by moving to a more powerful pda formalism for a (finite) regular language. Without more specific knowledge of the "instructional repertoire" of the human cognitive machinery, it would be quite difficult to sustain the argument that an algorithm

7. In general, a real-time Turing machine recognizing a regular language will read its input at a constant rate, e.g., k symbols at a time. Thus the demand that recognition be completed in time exactly $|n|$ requires that we be able to "re-program" the finite-state table portion of the Turing machine so that k basic instructions are "speeded up" into one instruction. One could use the standard method of Hartmanis and Stearns [1965] to accomplish this.

running in time |n| would be superior to one running in time kn, for small k.[8] We may conclude then that at least in the case of the trade-off between a finite-state and a push-down automaton (context-free) representation for the same (finite) regular language the potential loss in terms of increased recognition time complexity is not necessarily damaging. We can state the conclusion from a slightly different point of view. Naively, one might expect there to be a direct, one-for-one tradeoff between grammar size and recognition complexity, with grammar size denoting "space" and recognition complexity "time". From this crude classification it perhaps follows that whenever grammar size is increased, recognition time is decreased, and vice-versa. There are two problems with this rough assessment, however. First, grammar size interacts with recognition complexity, and so the trade-off is not direct. Second, the trade-off itself need not be one-for-one; as we have seen, we can decrease grammar size exponentially while increasing by a constant factor the time required for recognition.

Let us now consider the case of push-down automata vs. finite state representations for the same infinite regular language. The reason that finite state automata cannot represent some finite regular languages succinctly is that finite state devices have no "memory" besides what possible with a finite number of states. In particular, finite state devices cannot succinctly represent nested tree structure; they must in effect "memorize" it. In particular, a finite state device requires $O(2^n)$ states to "memorize" the language associated with a binary tree structure of depth n. The number of states demanded rapidly increases if, in addition, the finite state device must also *process* the binary tree it has stored, rather than merely recognize the terminal strings that the tree structure generates. To demonstrate this, we adapt another example of Meyer and Fischer's. Let us construct a hypothetical (and presumably unnatural) language whose string well-formedness conditions reflect a particular additional constraint on proper tree structure. The example is contrived, but serves to illustrate how a doubly-exponential increase in finite state machine size can occur. The artificial language has the following alphabet symbols: PP, V, V_1, and V_2 (two types of "Verbs"), and n different NP "types", NP_1 through NP_n. Strings in the language consist of an arbitrary number of NP_i's, $i > 1$, PP's, and V's, ending in an NP_1 and then followed by a string of V_1's and V_2's of length n, e.g., for $n=3$, there are three different NP's:

---

8. There is one additional complication, however. As Harrison [1978] observes, since it is possible for a linear-time algorithm to make an unbounded number of transitions proportional to the length of the input string without reading any input or after having finished reading all inputs and still be linear time, it follows that there could be an additional distinction in on-line sentence processing between real-time and linear time algorithms: a linear time procedure might demand processing "pauses" of length proportional to n at certain places in sentence processing or at the end of sentences. Note that a real-time algorithm does not permit unbounded delays of this kind. Detecting these differences experimentally might be quite difficult, however. Interestingly, the kinds of on-line psycholinguistic experimentation that has been done (see, for example, Kaplan [1974]) points to processing loads that do rise and fall with sentence length -- thus pointing to linear time, rather than real time, processing behavior in people. The entire matter is far from clear: the discussion in the psycholinguistic literature usually conflates the terms "on-line", "real-time", and "linear-time", making it extremely difficult to determine exactly what properties of human sentence processing are being measured.

$$PP\ NP_4\ NP_2\ V\ NP_3\ PP\ PP\ V\ NP_2\ V\ NP_3\ PP\ NP_1\ V_1\ V_1\ V_2$$

The language is further constrained in a complicated way as follows. Each V_i is to be thought of as a kind of "Verb" that intrinsically "marks" an NP_i to its left. (Intuitively, this "marking" process may be thought of as case marking, although it does not quite follow all the principles of case marking.) In addition, the V_i's differ with respect to the number of NP_i's they must mark; V_1 must mark one NP_i, and V_2 must mark two. Further, each V_i defines a domain of government over which it marks its NP's, according to the following definition: the first V_i (in this case, V_1) governs all the NP's up to the first NP_2. Since V_1 ignores all other NP's up to the first NP_2, we may think of this process as singling out a specific type of NP for marking. The second V_i is blocked from governing the NP's within the domain of the first V_i, from NP_1 up to the last NP that the first V_i marks, NP_2. The second V_i, again a V_1, again must mark just one NP, but this time must pick out the first NP_3 in its governing domain.[9] Finally, the last V_i, V_2, attempts to mark two NP_4's in its domain, which extends from the last NP marked by the second V_i to the left. If after the list of V_i's is exhausted the symbol immediately *after* the last marked NP_i is a V, then the string is in the language. If the (case) marking procedure for V_i fails to find the right number of NP_i's in its domain, or if the marking procedure ends with other than a V to the left of the last marked NP, then the string is not in the language. For example, the string above is *not* in the language so defined, because the last V_i demands that there be two NP_4's to mark in the domain extending from the left-most NP_3 (marked by the second V_i) leftwards. But there is only one NP_4. On the other hand, if we removed the last V_i from the string, then the resulting truncated string would be in the language: the second V_i would mark the left-most NP_3, the string of V_i's would be exhausted, and the next item to follow is a V, as required.

What we have done is construct a less artificial version of a regular language that Meyer and Fischer show can be recognized only by a finite state automaton with at least $O(2^{2^n})$ states (and hence is of least this size). In contrast, a deterministic push-down automaton of size $O(n^3)$ can recognize this language. (The push-down machine simply stacks the input string till it gets to the first NP_1. It then looks at the next symbol. If it is not a V_i, it rejects. If it is a V_1, it pops the stack until it gets to the first NP_2; if the next symbol is a V_2, it pops the stack until it gets to a second occurrence of NP_2. Thus all stack symbols from the point where NP_1 is first encountered until the first or second NP_2, counting from the right, are erased. If it cannot find such occurrences of NP_2, the machine rejects the string. It then reads the next input symbol and continues in this fashion. If after all the input symbols are exhausted the top symbol on the stack is a one, the machine accepts; otherwise, it rejects. The machine has n+2 stack symbols, n states, and an input alphabet of size n, hence is of roughly cubic size.)

---

9. Note that it is this property that does not seem to be a "natural" one; why should a Verb look for an NP of a certain sort in one domain, but not in another?

Why does the finite state machine recognizing this language require so many states? We sketch an intuitive version of the proof here. Meyer and Fischer show that the infinite regular language described above can be conjoined with a finite regular language describing ternary trees of depth n with two non-terminals and one terminals at each level to simulate the selection of a particular branch of a tree with a V at the end. A finite machine that can do this (1) must be able to recognize trees of depth n, which takes $O(2^n)$ states, as we have seen; and (2) must be able to store in its finite control all possible branching paths of those trees, since it must "remember", as it processes the terminal string of such a tree, whether there was a V, or a PP in each position of the string. That is, it must remember every possible tree structure and corresponding pattern of terminal strings. But since every terminal item that must be remembered can be either a V or a PP, and there are 2^n possible terminal items for a tree of depth n, there are 2^{2^n} possible patterns of V's and PP's to remember. But since this language taking a doubly exponential size finite state machine can be defined as the intersection of the ternary tree language, taking only exponential states, and the complex infinite language described above, it must be the case that the complex infinite regular language requires a doubly exponential size finite state machine. (Recall that to build a machine that mimics the intersection of two finite state machines at worst requires the Cartesian product of the two machines. Thus the doubly exponential size of the resulting language could not be due to the intersection alone; it must be due to the size of one or the other of the regular languages that form the intersection.)

In contrast, a push-down automaton can "simulate" the search for the right branch of a tree much more quickly, because (1) it can store the 2^n possible terminal elements using a representation of size just n; and (2) it can search this binary tree in polynomial time (via the complex infinite regular language described above).

In general, then, requiring a finite state machine to do the work of a push-down automaton can be quite expensive in terms of the number of states required. Just storing the representation of a tree structure demands an exponential number of states; if in addition one must compute paths through that tree, one must laboriously store all those paths, resulting in a further exponential explosion in the number of states that are required.

If we move to the case of other representations for context-free languages besides push-down automata, then Meyer and Fischer have shown that succinctness gains can be even larger. In fact, the potential gain in economy of description of a context-free grammar over a finite-state representation is not bounded by any recursive function. However, the example languages for which this compression result holds do not seem to be of immediate relevance for natural languages; the result depends upon the well-known fact that the non-accepting (invalid) configurations of a Turing Machine computation themselves form a context-free language, an "unnatural" language in this context. (See, e.g., Hopcroft and Ullman [1979]) As a result, we will not pursue the application of this theorem here.

It is also worthwhile to consider possible trade-offs between alternative representations even *within* a single generative class. For example, it is known that for some languages a non-deterministic finite state automaton is roughly exponentially more succinct than a deterministic finite automaton for the same language.[10] A well-known example of such a language consisting of all strings of length n that must have a distinct symbol, say, a "1", some number of tokens from the *end* of the string, say six. A NDFA with of size about n+2 can guess that a string will end in six symbols, but the corresponding DFA must simulate all these guesses, again exploding the number of required states exponentially in the number of tokens from the end that must be "remembered." But are any such examples linguistically relevant? As Chomsky [1965] has observed, natural languages do not appear to make use of co-occurrence restrictions that "count" words; for example, there is no natural language where the third word in every sentence must be, say, a Verb.[11]

Nonetheless, there are some apparent natural language cases where the advantage of a non-deterministic finite-state representation over a deterministic one is relevant. A particular example that arises in the parsing of extended context-free grammars as developed by Joshi, Levy, and Yuch [1980] will be treated in detail below. Briefly, Joshi *et. al.* show that a non-deterministic representation of tree structures can be exponentially more compact than a corresponding deterministic representation; this compact representation is then used to obtain a fast recognition algorithm for extended phrase structure grammars. Just as in the case of the finite-state/pda tradeoff, there is an additional cost in terms of recognition complexity if one adopts the more pwoerful representation. To test a sentence for membership in a non-deterministic finite state language with n transitions will take about $O(n^3)$ time.[12] But since the non-deterministic representation can be exponentially smaller than the corresponding deterministic representation, the change sometimes pays off; we shall provide a specific example of such a case below. Since this use of non-deterministic representations can be used

---

10. Note that the usual subset construction for converting a non-deterministic finite state automaton to an equivalent deterministic automaton demonstrates only that an exponential number of states in the DFA are sufficient, not necessary.

11. What accounts for the apparent absence of "counting" in natural languages? There are several possibilities. First, one might argue that potential exponential state-expansion selects against non-deterministic representations of linguistic knowledge. But this argument is actually backwards. Suppose (counterfactually) that linguistic knowledge can be represented as some regular set, and has a non-deterministic representation. As established above, this representation is at least as compact any deterministic representation, and for some regular sets is exponentially more compact. Perhaps, then, a non-deterministic representation entails some loss in another "functional" respect, say, in parsing efficiency. As we remark below, while this is true, there need not be any perceptible loss in parsing time: a non-deterministic finite state automaton need take only time $k\,n^3$ to analyze sentences of length *n*. If *k* is small, then the loss need not be great. In contrast, as we have seen, the fact that a counting language demands an exponential number of deterministic states for storage *might* prove to bear on the functional demand of acquisition. Interestingly enough, there is a theory of grammatical inference from positive-only evidence for the non-counting context-free languages developed by Crespi-Reghizzi [19/1]; we discuss this theory briefly in Chapter 3.

Finally, it is not altogether clear that counting is completely absent from natural languages. For example, some formulations of the Subjacency constraint require that one be able to "count" the number of "S" nodes that have crossed by a movement rule.

12. The method used is the same dynamic programming technique used to parse context-free languages: e.g., the CKY algorithm.

to handle context-free constructions in natural language that are describable via the extended phrase structure systems of Joshi *et. al.*, we conclude that it is sometimes advantageous to use a non-deterministic, rather than a deterministic, description of a linguistic system.

4.4 Succinctness of Extended Phrase structure grammars

Let us now turn from the simple case of finite state -- pda representations to a more realistic setting. Recently, Gazdar [1981] has proposed that the theory of transformational grammar should be replaced by a theory of extended phrase structure. Among the reasons he advances is the lack of putatively non-context free constructions in natural languages. Therefore, he argues, since transformational systems generate non-context free languages, the theory of TG is descriptively too rich; one would do better by adopting a theory with less expressive power, e.g., a theory based on formalisms that can generate only context-free languages. But as we just saw in the case of finite state vs. pda representations of self-embedding constructions, it might be the case that a more expressive system does a better job of capturing the correct generalizations about a set of surface strings than a less expressive system, even if that set is "only" finite state or context-free. We show below that Gazdar's extended phrase structure grammar must adopt a set of stipulations to describe the surface facts explained by the Government-Binding (GB) theory (Chomsky [1981]) theory exactly in the sense that a finite state machine must stipulatively simulate certain push-down automata.

To begin, we must first describe the potential context-free/non-context free representational tradeoffs, exactly as in the finite state -- context free case. The Gazdar theory is grounded upon formal results obtained by Joshi and Levy [1977b], originally developed by Peters and Ritchie [1969]. Let us briefly review this work.

The main result of Joshi and Levy is to show that a "locally" context-sensitive re-write grammar (local CSG) with rules of the form,

$A \Rightarrow B / \Phi\_\_\Psi \qquad \Psi\_\_\_\Phi$ a "proper analysis predicate"

(Here, the notation $A \Rightarrow B/\Phi\_\_\Psi$ means "re-write A as B in the environment Φ___greek(Y)".)

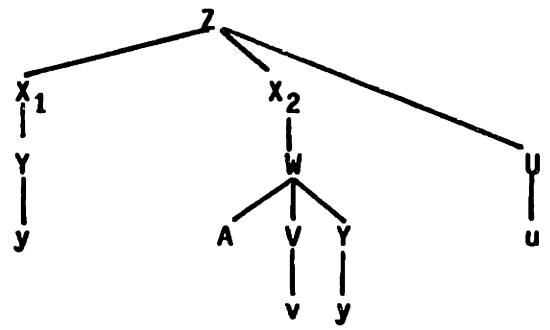
can generate only a context-free language.[13] Intuitively, a proper analysis predicate specifies a horizontal context "slice" through a tree that passes through the node being re-written (node A),

---

13. The inductive definition of a proper analysis of a tree *t*, P(t), is as follows: (i) if t = 0 (null tree), then $P(t) = \emptyset$; (ii) if t = root A dominating subtrees $t_0 \ldots t_n$, then $P(t) = \{A\} \cup P(t_0) \ldots P(t_n)$ -- i.e., the union of A with the concatenation of the proper analyses of the subtrees below A.)

touching one representative node from each subtree in the horizontal slice. For example, the cuts in (i) below are proper analysis paths, while those in (ii) are not:

(i)



example
proper
paths: Y, A, V, Y, U; X, A v, Y, u; y, A, v, y, U

(ii)

example
non-proper
paths: y, W, U (misses A); Y, A, V, U (misses Y subtree); Y, A, V, Y (misses U)

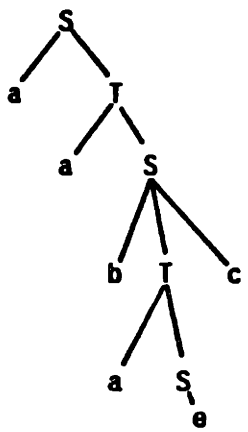
A proper analysis predicate rewriting A is then just a set of nodes Φ, Ψ, such that the path $p_1 \Phi A \Psi p_2$ is a proper analysis of the tree in question. Further, if the notion of a "slice" through a tree is extended to include cuts that pass from the root of a tree to a leaf, passing through a node to be rewritten, then this sort of predicate also generates only a context-free language.

Since the language generated by local CSGs is only context-free, for every local CSG there is an equivalent context-free grammar that generates the same language. In the spirit of the previous section we can ask the following question: Are there any context free languages such that local CSGs are more succinct than any (weakly) equivalent CFG for such a language? The answer is yes. To convert from a local CSG to an equivalent context free grammar, one must add new non-terminals that encode the context-checking work done by the context predicates. (Crucially, the amount of such context checking is finite, so that only a finite number of new non-terminals need be added.) Joshi, Levy, and Yuch [1980] illustrate the basic point with the following example. (We omit the demonstration that the predicate "dominates" is a local context predicate.)

the local CSG:

S⇒e S⇒bTc/a__∧T dominates S
S⇒aT T⇒bSc/a__∧S dominates T
T⇒aS

a sample tree:



An equivalent CFG:
S⇒e S⇒aT T⇒aS (original productions)

$S \Rightarrow aT_1$
$T_1 \Rightarrow bSc$

$T \Rightarrow aS_1$
$S_1 \Rightarrow bTc$

Two new non-terminals must be added along with two new productions in order to simulate the context-checking. The non-terminal S_1 represents ("remembers") the fact that T dominates S and that there is an *a* to the left; analogously for T_1. Note that the context predicates "a__∧ T dom S", "a__∧ S dom T" are eliminated, but reappear as new re-write rules. Assuming that these new rules take the same number of bits to write down as the predicates they replace, we see that the only difference between the two grammars is in the numbered subscripts for S and T. These will take 1 bit each of additional storage, for a total of 2 bits additional storage for the CF grammar as opposed to the local CSG (assuming a binary representation of subscripts). More generally, this result suggests that if we have q independent constraints of unit length or one constraint of lenth q in a local CSG then the equivalent CFG will take log(q) more storage. This intuition is confirmed by the following theorem

established by Joshi, Levy, and Yuch [1980]:

Theorem: (Joshi, Levy, and Yuch) *For any constant k, ∃ a local CSG G_1 s.t. G_1 is k times smaller than any weakly equivalent CFG.*

Proof: Given an alphabet Σ, the language of G_1 consists of strings of the form Σ *w*, where *w*= a single string of length q, $w=a_1...a_q$. The grammar to generate strings in Σ is simply $X \Rightarrow \sigma X$, for each σ in Σ; thus there are |Σ| such productions, each of constant length. The trailing suffix *w* is generated by the local CS rule, $X \Rightarrow a_q / a_1...a_{q-1}$. This last rule is of size O(q); if we make q large enough, then |Σ| is small compared to *q*, and $|G_1| = O(q)$.

Now consider an equivalent CFG for this language, G_2. It takes a q-state minimal machine to recognize L; that is, there must be *q* separate rules $X_i \Rightarrow X_j$, one for each state. Each state requires a representation of size O(log q) (in a binary representation), and there are q such states. Thus $|G_2| = O(q \log q)$. ∎

This result, however, is somewhat unnatural; the language consists of all strings over some alphabet followed by a fixed, special string of a certain length. Can we show that this succinctness result arises in a more natural linguistic setting, as we did for the finite state/pda case?

Again, the answer is yes: such a situation arises in the case of multiple so-called "filler--gap dependency" constructions, as in,

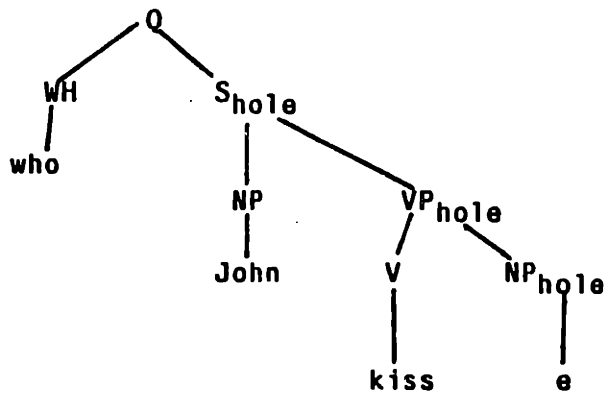
[Which violins]$_i$ are [these sonatas]$_j$ difficult to play [__]$_j$on [__]$_i$?

A system such as Gazdar's can easily *generate* such structures as these -- without subscripts indicating which filler is to be associated with which gap. The real problem, though, is that of co-indexing "fillers" and "gaps" correctly via an explanation rather than simply by listing the co-indexing pattern, a stipulation. Recall the formal definition of a stipulation: a description is a stipulation if it grows in size proportional to a list of what has to explained. In this case, what has to be explained is the observed pattern of co-indexing. We shall show that the augmented phrase structure system of Gazdar [1980] must necessarily grow exponentially in the size of the number of possible interacting filler-gap pairs in order to account for co-indexing; in contrast, a weakly equivalent theory that maintains a separate representational level to account for co-indexing (e.g., transformational theory) will increase in size only linearly. This exponential increase in grammar size corresponds to the fact that the phrase structure "solution" to the problem of long-distance dependencies is a pure stipulation, corresponding to the simulation of derivational history. (This result parallels exactly that for the finite state/pda case.) There is nothing, of course, to prevent one from extending the augmented phrase structure system by adding an additional level of representation at which co-indexing is checked for violations; Gazdar, for example, indicates that an unspecified "semantic" or "pragmatic" component could do

this work. This may well be true; it is a difficult proposal to evaluate given that no more detail has been given. Suppose it is true. Even then, as we shall see, the addition of an additional level of representation -- call it "pragmatic" if you will -- loosens the restrictions on generative power that were supposed to be a major advantage of a transformationless theory. In short, even though natural languages might be describable as context-free it is apparently the case that short descriptions of those languages demand a level of representation that goes beyond phrase structure and relates trees to trees via a relation that has the appearance of a rule *move α* -- exactly as has been argued since LSLT.

To account for the association between a fronted *wh* phrase and the "gap" from which it was moved, Gazdar introduces non-terminal subscripts, exactly as in the case of the forming an equivalent CF grammar from a local CS Joshi-Levy grammar. The subscripts record the "history" of a derivation so that the fact that a *wh* phrase was expanded at the head of a sentence will trigger a corresponding expansion of NP as a phonologically empty element later on:

Who did John kiss ___?



So, for every "normal" rule of the form, e.g., VP⇒V NP, there must be a corresponding rule that records the fact that a "filler" was expanded higher up in the derivation tree. Note that this is exactly the Joshi-Levy machinery; we could have just as well written the rule expanding NP as [e] like this:

NP⇒e/WH NP V___∧VP dom NP

This CS rule expands into the equivalent CF system:

$S \Rightarrow Q\ S_1$ or $S \Rightarrow$ NP INFL VP
S_1 NP INFL VP_1
$VP_1 \Rightarrow V\ NP_1$
$NP_1 \Rightarrow e$
etc.

There is another possibility for expansion, of course: the Subject NP could contain the *wh* trace. Thus, Gazdar must also include the rule $NP \Rightarrow e/WH\_\_\wedge S$ dom NP in his grammar. Plainly, this results in a large increase in the number non-terminals and productions required. Suppose that there are two or more displaced constituents within a sentence. Then, since the displaced elements can be in either the S or the VP, independently, there are 2 x 2 = 4 new local context rules that must be added (assuming that each gap must be distinguishable):[14]

	original rule: $[_\alpha\ X\ \ Y\ ]$
one filler:	$[_{\alpha/NP}\ X/NP\ \ Y\ ]\ [_{\alpha/NP}\ X\ \ Y/NP\ ]$
two fillers:	[X/NP1-NP2 Y][X/NP1 Y/NP2][X/NP2 Y/NP1][X Y/NP1 NP2]

Therefore, the size of this grammar will grow exponentially in the number of fillers and gaps that can interact -- a stipulative solution, since then $|G| \geq |S|$, where $|S|$ = the size of the set of possible trees.

How can this large number of non-terminals be reduced? There are basically two possibilities; we will show that either has the effect of implicitly adopting a transformational theory. First, one could directly add an additional representational level, beyond that of phrase structure, specifying the relationship between base generated phrase structure and phrase structure with constituents that have been displaced from their canonical argument positions. We return to discuss this alternative below. Second, one could advance an independent theory of constraints on rule expansions. Gazdar [1980] proposes one such extension, with the constraints on rule expansions taking the form of filters on expanded rules. For example, the A-over-A constraint is imposed by adding a filter of the form:

$$*[_{\alpha/\alpha}\ \ ]$$

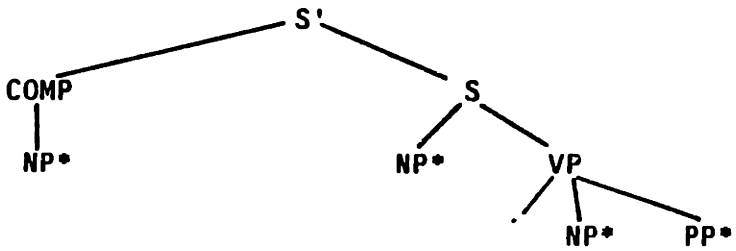
As Gazdar observes, Koster [1978] and Fodor [1980] have attempted to encode other sorts of constraints in phrase structure, in particular, the basic fact that moved constituents (or "fillers") must c-command their "gaps" and that the trace-antecedent relationship must obey Subjacency. Koster's idea was that the limited distribution of empty nodes could be described as restrictions on phrasal "projection paths":

---

14. If the two NPs are not distinguishable, then we reduce the number of context rules required so that the *total* number of context rules required grows as a Fibonnacci series in the number of fillers.

> In core grammar, empty nodes are only possible in positions immediately dominated by categories of the main projection (i.e. V^i or S^j; $3 \geq i,j \geq 0$). [page 570]

Thus the projection path for an English sentence will have the form:



Note that because NPs are branching, this constraint has roughly the same extension as a c-command restriction. (Indeed, part of the problem in evaluating constraints such as this is that in a limited domain of phrase structure there are many possible formulations that will have roughly the same extension.)

In the Gazdar framework, the projection path will be a list of nodes of the form $\alpha\beta$, where $\alpha =$ the X-bar projections of V or S and $\beta =$ NP. The projection path notion is then used by stating conditions on projection paths as filters on tree structures. For instance, Gazdar cites the work of Maling and Zaenen [1980] as an example of this use of projection path filters:

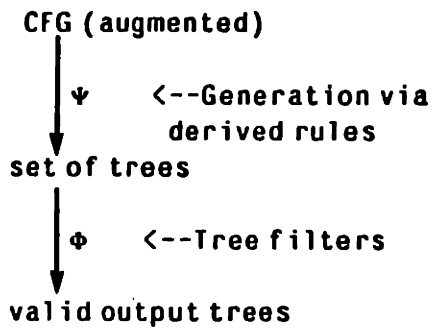
> Maling and Zaenen [1980] have proposed that the complex island facts in Italian discussed by Rizzi [1978] can be elegantly captured by the tree filter on projection paths shown in (10.10):
>
> (10.10) Throw out all trees containing a projection path of the form ...Q-bar/β...S-bar/β.
>
> [1980 page. 70]

(Q-bar is Gazdar's notation for an S-bar with a fronted wh-phrase.)

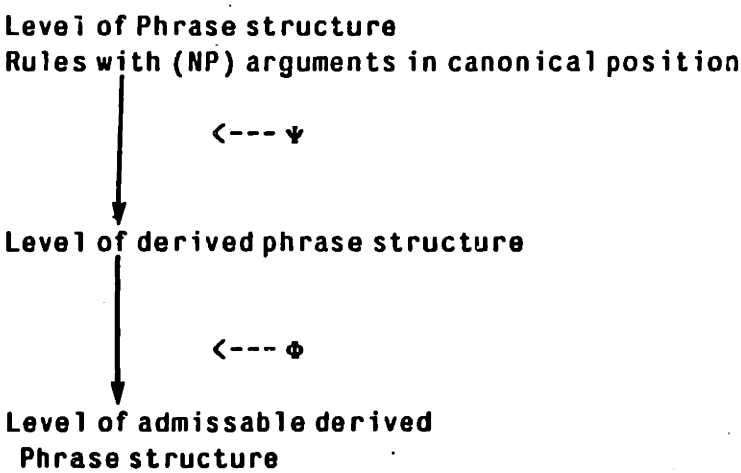
This filter simply "counts" possible bounding nodes (for Italian); if it encounters two of them, the tree is ruled out. Certainly this mechanism *describes* the Subjacency phenomenon in this particular case. How are the filters to be applied? The obvious proposal -- the one Gazdar makes -- is to apply the filters after the trees associated with the augmented phrase structure grammar are generated:

> It is worth remarking that imposition of tree filters of the type shown in (10.10) has no effect on the generative capacity of the class of grammars employing them: the output of a CF-PSG filtered by a finite list of such tree filters will be a CFL.
> [Ibid, p. 70]

Thus only certain trees will pass the output filters {F}. Mathematically then, we have the following: a set of phrase structure trees generated by the augmented phrase structure grammar -- let us denote the generation of these trees by the function symbol Ψ -- and a set of well-formed output trees (meeting the conditions of c-command, Subjacency, etc.) -- produced via the function specified by the tree filter set {F} -- let us denote this by the function symbol Φ:



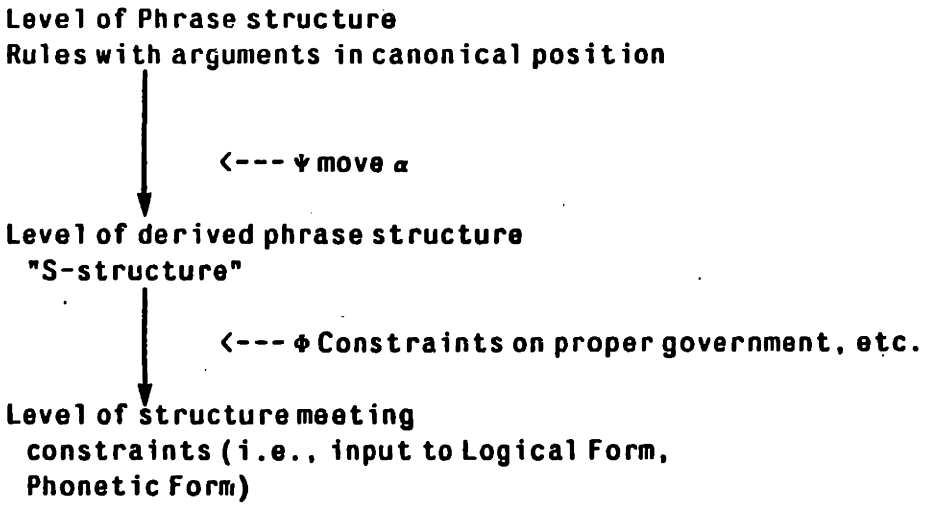
By definition, the set of valid output trees are constituent structures that meet the conditions demanded by the filter. Thus we have obtained, in effect, the following theory of grammar, with several levels of representation:



Why is the generation of admissable tree structures split into several steps in this fashion? Again we return to the theme of succinctness. The reason is that the list-like representation of the effect of possible movement rules is plainly inefficient. Since movement is assumed to be freely applicable, it is clear that a "shorter" description of the set of expanded phrase structure rules could be provided simply by mechanically generating them all via a single program schema that maps between the set of base structure rules and the resulting (observed) constituent structures:



How big is the program schema? All that is required is a single rule: *move* α, where α= NP or *wh*-phrase, in Gazdar's system. But this rule takes up only a *constant* amount of storage, no matter how many displaced constituents there are. The revised grammar would then consist of the following: (1) a level of base phrase structure rules; and (2) the single rule *move* α mapping to a level of derived phrase structure. Now suppose that at the level of structure derived by the single rule *move* α -- "S-structure" -- we impose the usual constraints of a modern transformational theory, e.g., Bounding constraints like Subjacency, constraints on proper government, and so forth. We have obtained the following theory of grammar:



In short, we obtain what is in essence the framework for a transformational grammar.

Note that since we *restricted* the form of the Gazdar derived rules in order to obtain a theory of the sort sketched in LSLT, namely, a mapping between the level of phrase structure and that of derived phrase structure via the rule *move* α, it is evidently true that the Gazdar theory without such restrictions is *more* general than the LSLT theory. That is, the Gazdar theory of grammar permits <u>any</u> connection between the level of phrase structure and the level of derived phrase structure, including <u>no</u> connection. Why is this? Besides "slashed category" labels, the Gazdar theory also admits a rule-to-rule mapping called a *meta-rule*. A meta-rule is simply an implicational relation of the form: IF <rule A> THEN <rule B>, where A and B are ordinary (normal node admissability or slashed category) rules. For example, meta-rules are used to describe the active-passive relation, so that for

every active phrase structure form, there is, predictably, a corresponding passive phrase structure form. The problem is that <rule A> and <rule B> can be *any* node-admissability conditions, hence any set of phrase structure trees, whatsoever. Given any two sets of phrase markers, one can write a "meta-rule" that connects them, no matter what the phrase markers look like, because there are no restrictions on meta-rules. If allowed full generality then, meta-rules permit one to associate levels of phrase markers randomly; there need be no *systematic* connection between one phrase structure expression and its derived constituent structure and the next phrase structure expression and its derived constituent structure. This is the worst possible situation, from an explanatory standpoint. That this is possible follows from the ability to write meta-rules indexed according to, e.g., particular verbs. Thus the derived constituent structure for "Mary loves Bill" could be different from, e.g., "Bill loves Mary" or "Mary likes Bill". Of course, the meta-rules actually written will not have such a random character, precisely because there *is* a systematic relationship between the level of base phrase structure and derived constituent structure, that of *move* α. Of course, one could say that one must write meta-rules that have the effect of systematically connecting two sets of phrase markers, according to the regularities evidenced by the rule Move-α, but to say this much is just to reinstate transformational theory once again.[15]

In then end then, the extended phrase structure approach simply reduplicates in another notation the discoveries of transformational theory. For example, note that the restriction that a trace must be at a one-node remove from an X-bar projection path ensures that a "filler", α, acting like an operator, will c-command its trace, γ. (Assume that c-command is defined as, e.g., α c-commands γ iff the first branching category that dominates α dominates γ.) Further, since α and γ are assumed to be in the same X-bar projection path, it also follows that Φ a maximal projection dominating α also dominates γ. But this is not enough to describe the distribution of $[_{NP}\ e]$, as shown in Chomsky [1981]. Both the subject and object of verbs meet this condition, yet, as is well-known, there is an assymmetry between the subject and object of verbs with respect to permissible violations of what has been called the "residue of the nominative island constraint" in Chomsky [1981]:

Extraction blocked from object:
*It is unclear $[_{Comp}$ what$_j$ $[_{Comp}$ who$_i$ $[\ t_i$ saw $t_j]\ ]$

Extraction permitted from subject:
It is unclear [who$_i$ [t_i saw what]]

Further, this constraint holds of nominatives but not of subjects of infinitives:

---

15. As noted by Chomsky, LSLT used the existence of a systematic connection between levels to argue for derived constituent structure. In this respect, the meta-rule approach does not even approach the explanatory adequacy of the theory advanced in LSLT.

*I don't know who would be happy if who won the prize. (tensed)
I don't remember who believes whom to have won the prize.

Of course, these facts can be accomodated in an extended phrase structure system by description: one simply encodes the relevant filtering effects on the phrase structure rules, with one filtering principle to handle the subject-verb assymmetry, and one to handle the tensed-infinitive assymetry. But this solution misses the point that these two constraints are correlated: apparently, wherever the assymmetry between subject and objects is found, so too is an assymmetry between tensed and infinitive clauses. It should, then, be possible to collapse these two principles into a single, "deeper" principle that is shorter than, hence explains, these two descriptive filters.[16] In fact, there is such a principle: as Chomsky [1981] shows, the observed pattern of constraints follows from the *Empty Category Principle*: $[_{NP}\ e]$ must be *governed* where the notion of *government* is defined as follows:

> In the structure $[_{\beta}\ ...\gamma...\alpha...\gamma]$, where
> $\alpha = X^0$ (α is a lexical item);
> α, γ are in the same maximal projection;
> then α <u>governs</u> γ.
>
> [1981 page 250]

Thus by adding the principle that a trace must be locally controlled by an X^0 category to a modified "projection path" constraint,[17] we obtain a more compact account of the distribution of trace. This example shows that one could simply absorb the sub-theories of the GB theory into an extended phrase structure system -- given that we have already noted the need for additional constraint. But then the point of the exercise is not clear; we will have simply obtained the GB theory translated into another notation.[18]

In this light, Gazdar's observation about the context-freeness of the structures generated by such a system is perhaps of mathematical interest, but it is not really relevant to the question of descriptive or explanatory adequacy. If it is true that "projection path" constraints can be "implemented" by means

---

16. We return to a formalization of this situation below.
17. As Gazdar [1980] notes, the projection path proposal of Fodor [1980] does not always yield the right results because it fails to incorporate the apparent blocking effect of maximal projections (such as VP) on government: according to Fodor's proposal, the last gap in "Who was it *t* that John saw *t'* is "properly governed" because it lies on the projection path running from the lower VP up through the projection path of the matrix VP (created by the first trace), and then to *who*. (Note that there is no trace in COMP, on this account.) This unbroken projection path is caused by the accidental confluence of two projection paths. This possibility is ruled out by the maximal projection constraint on government advanced in the GB theory. Of course, if we adopt this "local" account of government -- an account that also has computational advantages -- then we must also adopt something like the local government principles of the GB theory, rather than "long-distance" projection paths.
18. This same point can be made with regard to the problem of indexing traces with the operators that c-command them, a matter that is not considered in Gazdar [1980]. We shall discuss this problem in more detail below.

of tree automata that do not alter generative capacity beyond the context-free languages, then that same implementation method could also be applied to s-structures. (Indeed, one way to view the Gazdar system is as a kind of "analyzer" or "parser" for sentences, rather than as a theory of grammar. In essence, Gazdar's grammar simulates the states of a parser as it moves through a string, left-to-right, checking for well-formedness.)[19]

In sum, Gazdar's phrase structure expansions are simply a way of writing down all possible ways that the rule Move α could have applied to a set of base phrase structures, assuming that Move α applies freely; the post-movement constraints mimic (though in a rough way) the s-structure constraints of, say, the GB theory. (We have yet to indicate how other constraints of the GB theory, e.g., constraints on binding, might be incorporated into such a framework.) Gazdar himself claims that this expansion of rules can be handled by his version of a program schema, a "metarule". But are metarules any different from, say, a rule like *move* α? Recall that a Gazdar rule specifies a *node admissability condition* in the sense of Peters and Ritchie [1969], that is, a Joshi-Levy local context predicate that is true or false of a tree. Therefore each Gazdar rule specifies an admissable set of trees. A metarule associates a set of new context predicates -- hence admissable trees -- with a particular set of input admissable trees. Thus a metarule maps trees to trees.[20] The metarule that operates on a base phrase structure rule to distribute the category α among all its subconstituents is therefore just the rule *move* α, plain and simple.

What then of the question of efficient parsability? One of the chief benefits that Gazdar claims for his extended phrase structure system is that

> context-free languages are parsable in time proportional to the cube of sentence length or less...but no such restrictive result is known for the languages generated by transformational systems...
> [1981, p. 155]

Recall, however, that the algorithm on which Gazdar is basing his claim is the Earley algorithm [1968], and a more exact analysis of the running time of this algorithm is $k\ f(|G|)\ n^3$, where $|G|$= grammar size, in terms of number of rules, and n=length of input sentences. But as we have just seen, $|G|$ grows exponentially in in the number of interacting fillers and gaps in Gazdar's system, assuming that extended phrase structure rules are actually expanded and that projection path constraints are imposed *after* rule expansion. If so, then parsing time will also be exponential in the number of fillers, hence

---

19. An important consideration in evaluating a tree automata "filtering" approach has to do with the "size" of the finite state tree automata required to implement, say, Subjacency. We shall discuss this matter in more detail below.

20. This contradicts what Gazdar asserts in Gazdar [1980] page 43: "a transformation maps trees to trees whereas a metarule maps rules into rules".

exponential in the length of sentences. [21] But also recall that a modestly-restricted *Aspects*-style TG theory may be shown to generate exactly the languages recognizable in exponential time. Hence there is no apparent corresponding gain in recognition time in adopting an extended phrase structure system that directly incorporates the Earley algorithm.

Joshi, Levy, and Yueh [1980] take a different approach to this problem. Instead of using an augmented phrase structure grammar and directly applying the Earley algorithm, they retain a local CS grammar with its local context predicates and calculate possible expanded rules during parsing itself. Crucially, they find that computing proper analysis membership need take only time n^3, obviating the exponential expansion of the previous approach.

It is worthwhile to study the approach they use to achieve a sub-exponential time bound for calculating proper analysis membership, because it sheds some light on the meaning of the succinctness results themselves. First observe that the number of proper analysis trees for a given phrase structure tree grows exponentially in the size of that tree, in general. Therefore, any algorithm that calculates proper analysis predicates by holding an explicit list of these trees must necessarily take at least exponential time to execute. But as we have already noted, a list can be a very inefficient, space (hence time) consuming representation. Bunneman and Levy [1978] offer an alternative algorithm. They note that the set of possible proper analyses for a given tree of depth O(n) may be represented as a non-deterministic finite-state automaton with $\underline{n}$ states that accepts strings representing precisely the proper analysis paths. The transitions of the non-deterministic automaton denote possible choices for a proper analysis -- at each node, one decides to either accept that node as part of a proper analysis, or else to expand that node into its children as candidates for the proper analysis path. Thus, either S is a proper analysis path, and nothing else is; or else, the nodes immediately dominated by S (X and Y, say) are in the proper analysis path and nothing else it; or else, the children of X are in the proper analysis and Y is; and so forth. The non-deterministic finite automaton so constructed can be thought of as a graph with $\underline{e}$ edges; checking whether a given string is a proper analysis amounts to traversing this graph, a process that can be shown to take at most e^3 time. Thus the 2^n possible proper analyses can be stored in an automaton of size just $\underline{n}$, and the derivations of this automaton can be checked (as in the Earley algorithm for the more general case of context-free language recognition) in just time n^3.

---

21. This will be so only if we can construct a case where sentence length is polynomially proportional to the number of fillers – not an impossible situation, since we need only add a fixed number of lexical items, say, one verb and one NP Subject for each filler to obtain a full sentence with a large number fillers.
We might be able to adopt a "projection path" approach that can impose Koster or Fodor's empty node constraints "on-line". If so, then we can also use this approach in a move-α based parsing system as well.

In short, the proper analysis trees need not be explicitly stored, but rather computed as we go along. [22] Given this subprocedure for computing proper analysis membership, their algorithm runs in time at worst $k\ n^6$, where k is again a function of grammar size.[23] A proper comparison of this method thus demands a calculation of the size of a local CSG for natural, as opposed to artificial languages.

Finally, there is one more problem to deal with in augmented phrase structure systems, that of *co-indexing*. Consider a sentence such as, "Who did Bill kiss?". Not only is there a gap after *kiss* -- this being simply a re-statement of an obvious descriptive fact about this sentence -- but *who* is specifically associated with the "position" after *kiss*. This association is required in order to properly interpret the sentence. In the current theory of tranformational theory, the association of *who* with the null phonetic element *e* is carried out by the rules of binding. However, there are no such rules in the bare Gazdar phrase structure grammar. How then is this association to be indicated? The only possible solution that retains just phrase structure machinery is to subscript the fronted *wh* phrase with an integer, say i, propagate that index, and expand NP_i correpondingly as e_i.

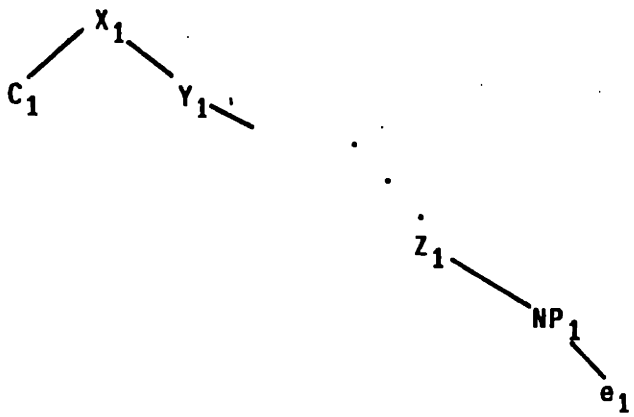
If there is only one possible displaced constituent and associated "hole" this solution is perfectly satisfactory, though unilluminating. The only "cost" is an increase in the number of non-terminals, proportional to the length of the non-terminal chain between displaced constituent and hole with which the constituent must be ultimately associated:

---

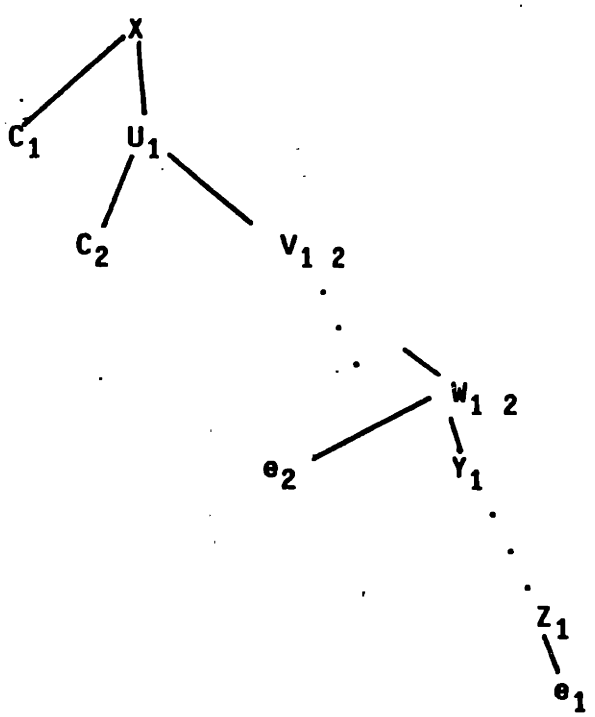
22. This is, incidentally, another example where computation can fruitfully replace simply "looking up" the answer to a problem. The automaton representation provides a much more compact way to store proper analysis paths, even though it demands extra computation time. In fact, this succinctness result is mathematically analogous to the finite state-context free tradeoff discussed earlier.

As noted in the text, Earley's algorithm shows that this result about storing proper analysis trees carries over more generally to the case of context-free grammar representations of sets: there can be an exponential number of derivation trees for a given context-free grammar, yet these all can be stored in cubic space. Thus even if the representation of the proper analysis trees required a context-free grammar, checking membership might take only cubic time. An interesting topic for research would be to determine why this method of storing constraints fails in the case of strictly context-sensitive derivations. Intuitively, it is because in a strictly context-sensitive derivation one can re-write the context variables, resulting in complex dependency chains. This additional complexity is reflected in the complexity of derivation trees for strictly context-sensitive languages.

23. More precisely, since the length of candidate derivation rules using Earley's method grows on the order of n^2, and since each rule may take m^3 time to check, where m = the length of the longest path in a local constraint, the total time is $n^2 \times m^3 \leq O(n^6)$. For parsing, the grammar size factor is roughly $|G|^3$.



However, as is typical in purely descriptive solutions of this kind, serious problems arise when we begin to consider cases that are even slighly more complicated. What happens when there are two, three, or more displaced constituents that interact? We must somehow indicate which displaced constituent is linked to which empty position. Given two possible displaced constituents, C_1 and C_2, the first empty expansion could be bound to C_1 and the second to C_2, or vice-versa. Suppose the former holds. Then we have a derivation tree as follows:

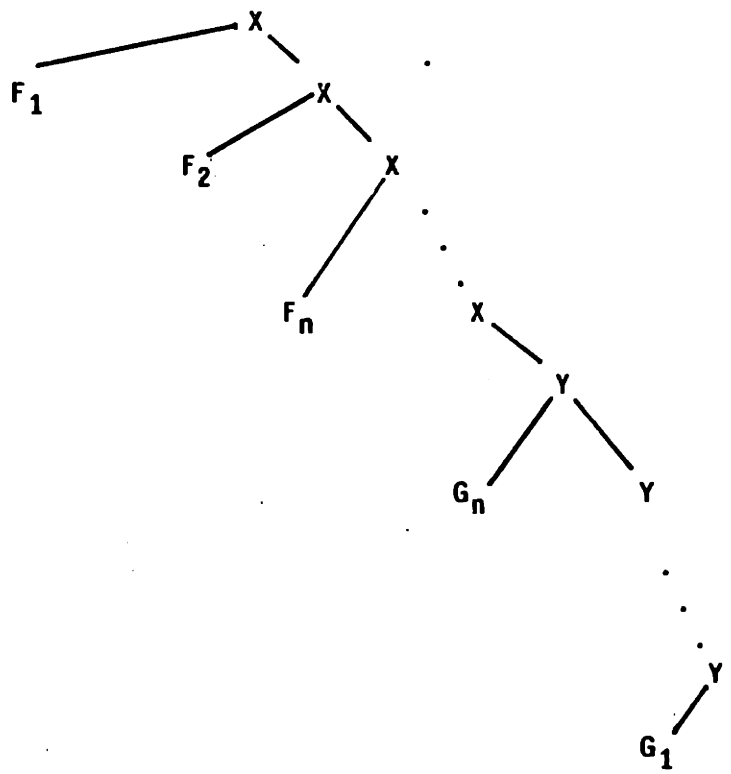


Note that we now have to add a rule to keep track of <u>two</u> displaced constituents; otherwise, we could

not distinguish between a sentence with two distinct displaced constituents and two empty argument positions, and an ill-formed sentence with two displaced constituents and only one empty argument position. We also have added a rule to "bind" the first hole e_1 to the first displaced constituent; the remaining displaced constituent is passed along, as required.

This grammar will generate sentences with nested filler-gap dependencies, such as, "Which violins are these sonatas difficult to play on?", but not the much less acceptable, "Which sonatas are these violins difficult to play on?". However, it is not *expanation* of this pattern, but rather only *description* of it. Why is this so? Observe first that the non-terminal co-indexing pattern could just have easily been intersected as nested; there is no reason why the expansion of W_{12} could not have been $e_1\ Y_2$, rather than the other way around.

But the problem cuts much deeper than this. How big will the extended phrase structure grammar have to be to handle n interacting dependencies? Consider what happens when the filler-gap co-indexing pattern is nested, as in the acceptable violins-sonatas example. In effect, the following pattern of fillers and gaps must be "recognized" by the local CSG, though the fillers and gaps are strung out along the phrase structure of the tree:



But this is simply the mirror-image language once again. The grammar must "remember" that fillers

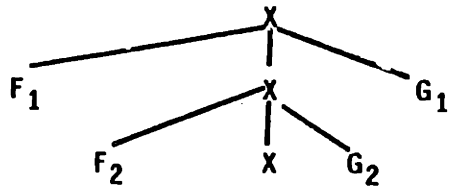
were expanded in the order $F_1,...,F_n$, and that gaps were encountered in the order $G_1,...,G_n$. Formally, as is demonstrated in Joshi and Levy's proof that local CSG's generate only context free languages, this means that the non-terminals of the grammar and its expansions rules simulate a finite-state tree automaton recognizing the mirror-image pattern $F_1...F_n\ G_n...G_n$. As we have seen, this will take a finite state machine with an exponential number of states -- another demonstration of the exponential grammar size increase discussed above.

We thus obtain the following:

Theorem 4.1. *Let L be a context-free language with n nested fillers and gaps generated by an associated grammar with right-linear branching structure. Then any extended phrase structure grammar for L will be of size at least* $O(2^n)$.

Proof. A tree automaton is a generalization of the usual finite state machine where the transition function is extended from a mapping on $Q \times \Sigma \rightarrow Q$ -- from states and alphabet symbols to a new state -- to $Q^n \times \Sigma \rightarrow Q$ -- from a set of states and alphabet symbols to a new state. Thus it may be visualized as "processing" a tree from daughter nodes to mother nodes: the daughter nodes represent the set of states Q^n, and the mother node the state after transition. Thus the usual deterministic finite state machine is a degenerate "tree" where all state nodes lie along a single straight line. Suppose that there exists a (deterministic) finite state tree automaton of size less than $O(2^n)$ that can accept exactly L with right-linear branching structure. By re-writing the F's and G's as terminal elements, it is clear that this tree automaton implies the existence of a deterministic finite state automaton of size less than $O(2^n)$ that can recognize the palindrome language, a contradiction. ∎

The reason for the huge increase in states demanded by the above example is that we are in effect not using the power of a context-free formalism -- as indicated by the branching structure of the tree above. Some researchers have attempted to circumvent this particular dilemma by assuming that the pattern of fillers and gaps follows a rigid nested pattern (hence, is not random; hence, ought to be describable by other than a simple list of filler-gap assignments). For example, Fodor [1978] has claimed that fillers and gaps are always co-ordinated so as to obey push-down stack discipline, thus "explaining" the violin-sonatas examples. This effect could be incorporated into a phrase structure model by relying upon the power of phrase structure rules to generate push-down structure:

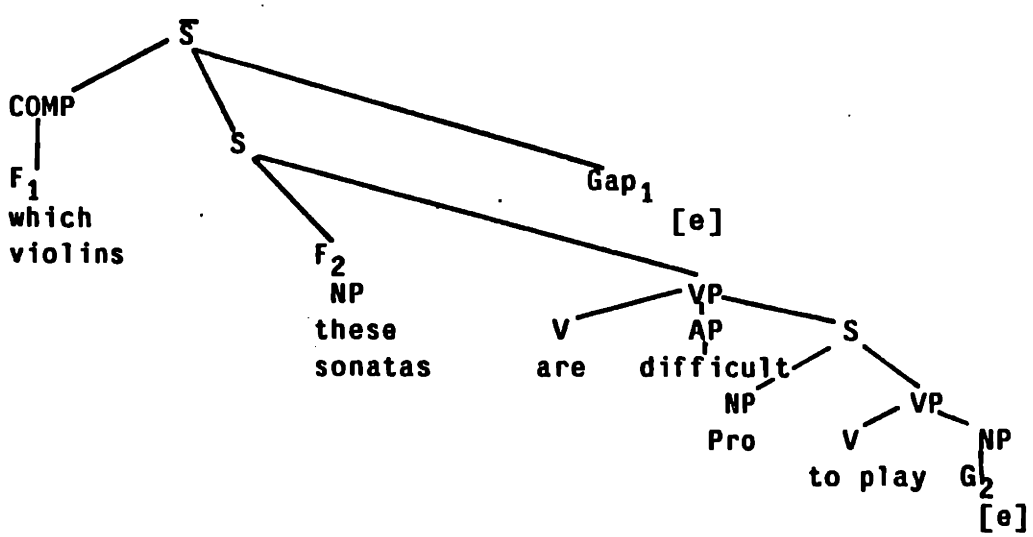


As we have seen, a push-down automaton of size but O(n) (assuming a fixed alphabet) could handle *n* filler-gap dependencies nested in this fashion. Fodor has attempted to add a functionally-motivated generalization about co-indexing to the Gazdar phrase structure system. It is this generalization that compresses the description of nested filler-gap co-ordination, hence serves as an "explanation" for it. (Note that the size of the pda description is smaller than a list of the trees so described.)

There are two problems with this solution. First, it is descriptively inadequate; as several writers have observed, there are cases where non-nested patterns are more acceptable than nested ones:

> *Which oven did the cake take you all day to help John bake in?
> (Stowell)

Second, the self-embedded rules demanded by nested dependencies lead to incorrect structural descriptions. For example, if the nested dependency co-ordinations were generated by context-free rules, then the structure of the *which violins--sonatas* sentence would be expected to have a nested structure something like the following:



In order to take advantage of the power of push-down machinery to associate gaps and fillers, we have

to associate Gap_1 with its filler *which violins* as the filler is expanded in COMP. But then, the gap is attached, incorrectly, to the highest S node. Similarly, the inner filler-gap should be expanded along with the S being re-written as the NP *these sonatas* and a VP, but this also results in the wrong phrase structure. One solution to this problem is Gazdar's -- expand a filler and gap together by marking via the "X/NP" notation the non-terminal along which the gap will be passed instead of actually expanding to an [e] terminal string at that point. However, this returns us to our original problem of excessively expanding the non-terminal set of the grammar; while it is true that for a fixed number of filler and gap pairs only a finite amount of information need be "passed down" the tree, as we have seen, the amount of such information can grow exponentially large in the number of fillers.[24]

Alternatively, one could, as Fodor [1978] seems to suggest, claim that the association of fillers and gaps is carried out by the language processor, in particular, by a push-down mechanism distinct from that associated with the phrase structure grammar, something like the *hold cell* mechanism of Augmented Transition Network parsers. (Woods [1970]) The model of sentence processing that is envisioned is roughly as follows: one has a "standard" automaton to handle basic phrase structure parsing, to which one adds a *separate* push-down store that holds fillers in a first-in, first-out basis. When a gap is encountered, the filler currently on top of this stack is automatically co-indexed with it. Since the phrase structure grammar generates the right structural descriptions, and the auxiliary push-down store performs the (assumed correct) nested co-indexing, all seems well. What, though, is the power of this augmented machine? If the auxiliary "hold cell" stack can store any kind of constituent phrase, then we have in effect the power to perform a (non-erasing) push-down transduction on the input language. By Fodor's assumption, the input language is generated by a context-free grammar. Thus we have a push-down transduction of a context-free language. But as is well known (see Ginsburg and Rose [1966] and also immediately below), the non-erasing push-down transductions of the context-free languages yields the context-sensitive languages -- an increase in generative power. Thus, by incorporating an auxiliary push-down stack to handle filler-gap dependencies, we have in effect added back the power that we thought we had just eliminated.

Summarizing, both proposed solutions to the problem of nested dependency co-indexing involve adding back descriptive devices, either by means of an exponential number of non-terminals or by an auxiliary push-down stack. By increasing the number of non-terminals we eliminate the possibility of obtaining an explanation of co-indexing; the size of the description of the data approaches the size of the data itself. By adding an auxiliary stack, we re-introduce the very generative power that we had sought to avoid in the first place. A push-down stack of depth n, as we observed earlier, has the power

---

24. Furthermore, these sentences do not seem to have the perceptual difficulty associated with center embedded sentences, contrary to what these structures suggest. Nor do they have the intonation patterns characteristic of a center-embedded constituent structure, etc.

to encode 2^n finite states in space just *n*. Further, any approach that adds a new representational level that operates on the trees produced by the syntactic component -- e.g., a level of semantic interpretation -- also has the potential to increase generative power in just this fashion.

Fodor's solution to co-indexing is thus to throw the explanatory burden from the grammar to the parser. But this approach is successful only if co-indexing actually nested, and this does not appear to be the case, as we saw in sentences (X) above. Pursuing the problem of co-indexing a step further, let us consider how to handle *nested* co-indexing patterns. If we assume, along with Fodor, that it is parsing machinery that results in a pattern of nested filler-gap dependencies, how are intersecting dependencies to be accounted for? Note that intersecting dependencies are permitted in English:

Who$_i$ did you$_j$ ask Gap$_i$ whether Gap$_j$ to blame yourself?

There are two subcases to deal with, depending upon whether there are a finite or an infinite number of such intersecting dependencies.

> (i) There are a finite number of intersecting dependencies.
> In this case, we can code the intersecting patterns via an expanded non-terminal set, as before. For example, the pattern, $F_1\ F_2\ F_3\ G_1\ G_2\ G_3$ can be encoded via the rules, $X \Rightarrow F_1\ Y_1$; $Y_1 \Rightarrow F_2\ W_{1\,2}$; $W_{1\,2} \Rightarrow F_3\ U_{1\,2\,3}$; $U_{1\,2\,3} \Rightarrow G_1\ V_{2\,3}$, etc. As before, though, there is an exponential number of new non-terminals and rules that are created.
>
> (ii) There are an infinite number of intersecting dependencies.
> In this case, it is well known that although a context-free grammar can generate the string set $a^n\ b^n$ (where a= NP fillers, b= [e]), it cannot correctly associate a_i with b_i in the pattern, $a_1 ... a_n\ b_1 ... b_n$, this being a strictly context-sensitive language. Thus, in this case we must move beyond the power of a local CSG. Note that the machinery advocated by Fodor would suffice to handle this situation (if we are allowed to place any sort of material on the auxiliary push-down stack).

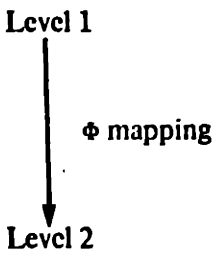
In either case (i) or (ii), we do not obtain an explantion of an intersecting pattern of dependencies -- we simply *describe* the pattern. (Indeed, in case (i) we *cannot* obtain explanatory power from the syntax alone, because, given 2^n non-terminals, we can describe any such pattern.)

To summarize, there are deep problems in using phrase structure systems to simply *encode* rather than *explain* linguistic data. Context-free or local CSG re-write rules are really a crude sort of "progamming language" for describing "surface" linguistic phenomena -- where "surface" phenomena are sets of acceptable trees. (The level of description is one step removed from simple surface strings, still, the empiricist emphasis is clear.) What is of interest is *not* the programming

language itself, but what the language can express, and how succinctly. The formalism, in and of itself, is of no value for explanation. But the cost in attempting to encode *all* such data using one sort of mathematical machinery can be, as we have seen, quite high. The attempt to collapse *all* generalizations into one level of representation does not succeed precisely where those generalizations seem to relate to different levels of representation. Just as phonological rules are best stated at the PF level, there seem to be rules -- like the Empty Category Principle or Subjacency -- that apply at the level of logical form or the level of s-structure. When an attempt to collapse two distinct representational levels fails, it does so in a distinct and detectable way. The rule system required to encode the effects of the additional representational level expands as the power set of the number of phenomena that used to be accounted for by the presumed superfluous additional level of representation. This kind of expansion is observed when we try to encode hierarchical structure into a finite state device, and it is also observed when we try to encode co-indexing or the rule of *move* α into an extended phrase structure system. In either case, we may conclude that an additional level of representation should be restored to the theory in order than it may retain its explanatory power.

4.5 A formal theory of inter-level modularity

Intuitively it is clear why adding additional representational levels should add extra power to a theory. Suppose that we have two levels of representation, L_1 and L_2, associated by a mapping Φ. (These levels could of the sort described in LSLT, e.g., P-markers.) The function Φ refers to the structure of level L_1, since L_1 serves as the domain of Φ. The extra power of this approach can be ascribed to the ability of Φ to look at the entire consituent structure of L_1 in constructing L_2; roughly, L_2 can make an "extra pass" through the structure L_1.



Let us make this intuition precise by developing a formal theory of inter-level modularity, one able to describe, for example, the theory of linguistic levels as outlined in LSLT. First, let us restrict Φ so that it is not just *any* mapping. There are three choices that will be of interest:

> (1) Φ is a generalized state machine (gsm) mapping.
> (A gsm is a generalized finite state machine that can emit any string, including the empty string, in a single move; formally, a 6 tuple: $(Q, \Sigma, \Delta, \delta, q_0, F)$; δ maps $Q \times \Sigma$ to subsets of $Q \times \Delta^*$; Q = states; Σ = input alphabet;

F = final states; q_0 = initial state; Δ = outut alphabet.)

(2) Φ is a push-down store transducer (pdt).
There are three sub-types of pdt that will be relevant.
(i) unrestricted; (ii) deterministic; (iii) non-erasing.

(3) Φ is a linear-bounded transducer (lba with output)

A formal version of our question about the power of representational levels can now be provided. Suppose that one representational level is expressed as a language L_1 (L_1 = regular, context-free, context-sensitive, etc.); level L_2 is expressed as a language as well. Note that in the usual formulation, a level is an algebra that includes a set of "P-markers". By regarding input and output levels as languages, we in effect taken P-markers to be not trees, but a set of strings -- as was done in LSLT and in Lasnik and Kupin [1977]. Let us say that the mapping associating level L_1 with L_2 is given by a transducer Φ. Now the question about the power of a two-level theory is reduced to the question: what range of output languages is produced by $\Phi(L_1)$?

Importantly, most of these questions have already been studied in the literature of formal language theory. Let us first consider a specific example.

Suppose that the input language is context-free and that Φ is a (possibly nondeterministic) pdt. Informally, suppose that the input string is of the form, say,

$$w_1 w_2 \ldots w_n$$

The pdt allows us to push elements of w onto a stack and retrieve them in reverse order. For example, assume that we stack elements $w_i,\ldots,w_j$, and then retrieve these elements, when reading, say, w_k. Note that we can now re-process w_i through w_k (though in reverse order). Thus a pdt allows us to make a kind of "second pass" through the input string, in reverse. That this second pass can result in an increase in generative power is established by the next example.

Example 1. Suppose that the input language is the context-free mirror-image language $w\, w^R$. Now we suppose that Φ on this input reads and outputs w_i's until it guesses that the middle of the string has been reached. It then stacks the remaining w_i's until it reaches the end of the input string. Finally it outputs the w_i's in its stack. Thus the language output by the transducer is $w\, w$ -- a strictly context-sensitive language.

We see, then, that transduction, the formalized version of the notion of two representational levels connected by a mapping, can greatly increase the power of one's theory in an automata theoretic sense. The mapping can even be quite restricted -- in this case, a push-down transduction -- an an increase in power can still be obtained. Results such as these have been studied extensively by Evey [1963],

Ginsburg and Rose [1966] and others. The following table summarizes the results from these sources.

Input Language, L_1	Transduction	Output language, $\Phi(L_1)$
regular	finite state	regular
"	gsm	"
regular	pdt	context-free [Evey]
regular	linear bounded	all r.e. sets [Ginsburg-Rose]
context-free	finite state	context-free
"	gsm	"
context-free	pdt	all r.e. sets
context-free	non-erasing pdt	context-sensitive
context-sensitive	finite state	context-sensitive
context-sensitive	gsm	context-sensitive
context-sensitive	non-erasing pdt	context-sensitive
context-sensitive	lba, non-erasing	context-sensitive

The reason that a pdt operating on a context-free language gives all the r.e. sets follows from the well-known fact (see, e.g., Hopcroft and Ullman [1979]) that the sequences of invalid computations of a Turing machine considered form a context-free language. Thus, it is possible to "check" any TM computation (expressed as a language) using a push-down stack transduction.[25]

Example 2. (Ginsburg and Rose, 1966). Suppose that the input language is the context-free language $a^i b^j c^j$. The pdt simply checks that $a^i = b^j$, and outputs the string if so. Hence the language output by Φ is $a^i b^i c^i$.

When is the language output by some pdt Φ still a context-free language? This question was answered by Evey [1963]: if the pdt subsumes the input language in the sense that all the transitions in the pda

---

25. In more detail, suppose that some derivation of a Type 0 grammar is as follows: $\sigma \rightarrow u_1\varepsilon_1 v_1 \rightarrow u_1 \beta v_1$ (via the rule $\varepsilon_1 \rightarrow \beta_1$)$= u_2 \varepsilon_2 v_2 \rightarrow u_2 \beta_2 v_2$ (via the rule $\varepsilon_2 \rightarrow \beta_2$), etc. The following context free language can be used to "check" this derivation: let S$= \{\sigma \# \beta^r \#$ s.t. $\sigma \rightarrow \beta$ is a production of the grammar$\}$; S denotes the "start" of a derivation, and "r" the reversal of a string. Let Y$= \{u\varepsilon \# \beta^r u^r \#$ s.t. u is in V^* and $\varepsilon \rightarrow \beta$ is a production of the grammar$\}$. Then the language Y A \$ (\$ a special terminator symbol) is a context-free language that checks derivations in the grammar.

required to generate the input language are also in the pdt, then the output language $\Phi(L_1)$ will also be context-free. For instance, the pdt's in both Example 1 and Example 2 do not meet this condition. In Example 1, the pdt does *not* have the rules to generate the input language $w\ w^R$, hence does not subsume the input language; similarly for Example 2.

The table indicates that a cascade of two or more pdt's is very powerful indeed. As Ginsburg and Rose note, since a single pdt can produce any context free language, and a non-erasing pdt of CFLs gives all the context-sensitive languages, a cascade of just two pdt's can yields any CSL (and two arbitrary pdt's hooked in tandem yield <u>any</u> r.e. set). It is this property that gives two (or multi) level theories their enormous potential succinctness, hence explanatory power, over single level theories.

The table of pdt transductions provides a summary of when we may expect a two-level theory to dominate a single-level theory. For example, consider a context-free language coupled to a non-erasing pdt. Suppose that the context-free language is describable by a grammar of size <u>m</u> and the pdt is of size <u>n</u>. Then the resulting theory is of size $O(m+n)$, and can describe any context-sensitive language. Now recall Fodor's [1978] proposal for describing filler-gap dependencies. As we observed, her processing proposal amounts precisely to a two-level theory, a pdt transduction of a context-free language. Now compare an extended phrase structure account of the case of crossed dependency structures. If there are an unbounded number of such dependencies, then a single level context-free theory cannot describe them all, and so is dominated by the two level theory. But if there are only a finite number of such dependencies, then, as we have seen, the size of the grammar is grows exponentially, and is hence also dominated by the two level theory. Let us make this notion of "dominate" precise with the following definition:

Definition. A theory A *dominates* a theory B for a family of languages $\mathcal{L}$ iff either
(i) A is descriptively adequate and B is descriptively inadequate; or
(ii) A and B are descriptively adequate, and
$\forall L \in \mathcal{L},\ D_A(L) \leq D_B(L)$

4.6 A formal theory of intra-level modularity

As pointed out earlier, besides the possibility of inter-level modularity, in the form of two or more representational levels, one can have *intra*-level modularity, in the form of several distinct, independent sub-components that interact at a single representational level to account for an array of surface data. For instance, in the GB theory we find the sub-theories of Binding, Bounding, and Case assignment operating at the level of s-structure, in contrast to the Case Filter (operating at the level of phonetic form) or the Empty Category Principle (operating at the level of logical form). Just as with inter-level modularity, it is a common intuition that one has advanced a better explanation of some

array of facts when they are accounted for by several small, independently operating principles.

Why should this be so? Again we sketch a formal analysis. Fix a single representation level, L_1. Suppose that L_1 is represented via some machine, M_1 (if L_1 were a context-free language, then M_1 could just be a push-down automaton). Let the size of M_1 be l. Suppose that a principle P_1, consisting of a finite list of strings of L_1 to be ruled out, is applied at representational level L_1. Let us represent this filtering effect of principle P_1 by the intersection of a finite automaton representing the complement of P_1 with whatever (machine) representation we have chosen for level L_1. Since P_1 can be represented as a regular set, the complement of P_1 can also be represented by a regular set, and is the language accepted by some machine, M_1, of size n. Thus the filtered output language is just L_1 intersect L_{M_1}. Similarly define machine M_2 for the complement of a second principle, P_2. Then the desired application of both principles can be computed via the intersection, L_{M_1} intersect L_{M_2}. By cascading the machines, the total size of this system is just the sizes of its subparts, e.g., $O(l+m+n)$.

Now consider replacing principles 1 and 2 with a single principle, P, that does the work of both of them. Since the replacement must be descriptively adequate, we must have that $L_{M_P} = L_{M_1}$ intersect L_{M_2}. That is, the machine for principle P is just the intersection of the (complement of) the machines for principles 1 and 2. Suppose that we construct this machine by the standard intersection construction for two finite state automata. This composite machine is just the Cartesian product of the two original machines:

Machine 1 = $(Q_1, \Sigma, \delta_1, q_1, F_1)$

Machine 2 = $(Q_2, \Sigma, \delta_2, q_2, F_2)$

Intersection =
$(Q_1 \times Q_2, \Sigma, \delta, [q_1\ q_2]\ F_1 \times F_2]$

where $\delta([p_1\ p_2], a) = [\delta_1(p_1,a), \delta_2(p_2, a)]$

That is, the size of the single new machine will be at most $O(n^2)$. When will it be less then this? The two extremes to consider are when two principles are perfectly correlated, and when they are perfectly independent. Suppose first that filtering principle P_2 is predictable from P_1, in the sense that whenever P_1 rejects a string, so does P_2. This means that the complement of $L(P_1) \subseteq$ complement of $L(P_2)$, or that $L(P_1) \supseteq L(P_2)$ But then, we can replace the finite state transition diagram representing the complement of P_2 with the machine for the complement of P_1. The composite machine's states $Q_1 \times Q_2$ and transitions are replaced by the states and transitions of the machine for the complement of P_1. Hence the composite machine is just the size of the complement of P_1. (We assume that both

finite state machines were reduced to begin with.) Similarly, if principle P_2 completely predicts principle P_1, then the size of the composite machine is just the size of the machine representing the complement of P_2. For example, consider the following two "principles" as represented by finite state machines P and Q. P accepts the strings $a\,a^*\,b$; Q accepts the strings $b^*\,a\,a^*\,b$. The intersection is just the the language of P.

At the other extreme, suppose that two principles are entirely uncorrelated, so that the accepting and rejecting pattern of either principle cannot be predicted given knowledge of the other principle. For example, consider two finite state machines P and Q. P accepts strings of the form a^{2n} and is of size two; Q accepts strings of the form a^{3n} and is of size three. The intersection language consists of strings of the form a^{6n}, and hence demands a machine of the size $2 \times 3 = 6$.

transitions for Machine P:
(p_0, a)-->p_1
(p_1, a)-->p_0
final state = p_0

transitions for Machine Q:
(q_0, a)-->q_1
(q_1, a)-->q_2
(q_2, a)-->q_0
final state = q_0

Composite machine:
$([p_0, q_0], a)$--> $[p_1, q_1]$
$([p_1, q_1], a)$--> $[p_0, q_2]$
$([p_0, q_2], a)$--> $[p_1, q_0]$
$([p_1, q_0], a)$--> $[p_0, q_1]$
$([p_0, q_1], a)$--> $[p_1, q_2]$
$([p_1, q_2], a)$--> $[p_0, q_0]$
final state = $[p_0, q_0]$

We conclude that if two principles that can be represented as finite state devices are truly independent, then collapsing them together, as expected, expands the size of the rule system required quadratically. If two principles are completely correlated, then one can collapse them without a concomitant increase in the size of the machine required to represent the principles; in fact, the size of the resulting machine is max {Size (P_1), Size(P_20} $\leq$ Size ($P_1 + P_2$). (Now construing a principle P as a filter, so that we really must consider the complement of P for the intersection.) Again, this result corresponds to the intuition that a principle that is completely deducible from another principle can be eliminated.

By extending this argument, we see that each additional independent principle can potentially simplify a grammar by a multiplicative factor. Thus, if there are m independent principles all of size n, then the total size of the rule system assuming independence will be roughly mn, but the size of a single-principle system that attempts to collapse the m principles into one can be n^m.

One can use similar techniques to investigate the succinctness gains possible when describing for what is typically called "complementary distribution." Consider the following pair of examples in Chomsky [1981]:

> They believe {*PRO/each other} to be intelligent. They know how {PRO/*each other} to be intelligent.

PRO and *each other* are said to be in "complementary distribution" because PRO can appear exactly where *each other* cannot, and vice-versa, at least at the level of s-structure. Suppose that we arrive at a principle P that accounts for the distribution of PRO and *each other* in the first sentence -- let us assume that P can be represented as a finite state device of size n that accepts the string, *they -- believe -- each other -- to be -- intelligent* and rejects the string, *they -- believe -- PRO -- to be intelligent.* Suppose that we attempted to account for the distribution of PRO and *each other* in the second sentence independently of the first -- that is, we ignored the complementary distribution. This would require the specification of a second machine, Q, of size let us say m. Presumably, m could be roughly equal to n, though this is not necessary. Then the total information that would have to be specified to account for the distribution of PRO and *each other* would be approximately m+n -- the sum of the specifications for the two sentences. But then, this analysis makes no assumptions at all about the complementarity of the second sentence; *any* distribution pattern of PRO and *each other* could be handled by a specification of this size. For example, the pattern, *they know how PRO/ each other to be intelligent* could also be accounted for by a specification of size m+n. Therefore, we have not explained the complementary pattern. Since all possible patterns can be described by a machine of size m+n, we have only stipulated the complementary distribution, not explained it. Recall that for our account to be explanatory, we must arrive at a description of the complementary pattern that is shorter than a mere list of the patterns themselves.

But such a "compressed" description *is* obtainable in this case. Since PRO and *each other* are in complementary distribution, we can use the machine for P to derive a machine Q that accounts for the observed distribution in the second sentence as follows: we take P and turn final states into non-final states and vice-versa.[26] If P accepts, then our new machine rejects; if P rejects; then our new machine accepts. There is, of course, one additional difference that distinguishes P from Q, and that is the

---

26. We assume a reduced canonical representation for finite-state machines, including at most one "dead" non-final state, and one final state.

context that triggers the complementary distribution in the first place, the context *know how*. But then, the specification of Q requires just one additional bit of information besides the specification required for P. (The way we have described the complementation process we will need in addition two new states, to lead from the old "dead" state to a new accept state, and from the old accept state to a new "dead" state.) Thus the amount of information now required to specify the distribution of PRO and *each other* has been reduced from m+n to n+c, c a small constant. Since n+c is less than n+m for m sufficiently large, we conclude that the new account is explanatory.[27]

---

27. Of course, as Chomsky observes, the actual distribution of PRO and *each other* is more complicated than this simple account would suggest. For one thing, even though PRO and *each other* are in apparent complementary distribution in the surface strings above, they share a common distribution sentences with verbs types different from the *believe* and *know* type:

We want {PRO/ each other} to win.

How should this additional complication be handled? Remaining neutral as to the details, one approach is to identify the *want* type verbs as a special class that triggers a common distribution of PRO and *each other*. But since the {PRO/each other} distribution is not predictable from a {*PRO/each other} pattern, the machine we need to handle the shared distribution will again be of size m (probably close to n). This is the approach taken by Gazdar [1981], for example.

There is, however, another approach. We can retain apparent complementary distribution if we assume that this distribution holds, not at surface structure, but at s-structure, and if we take the s-structure of the *want* sentences to be as follows:

(i) We want [PRO to win]
(ii) We want [for [each other to win]]

5. Locality Principles and Acquisition

The acquisition procedure presented in this thesis is based on a parsing model, Parsifal. Chapter 3 showed, informally, that the acquisition procedure and its parser are related in the following way: if a grammar G is local parsable (in the sense of Marcus), then G is learnable via a "k-tail" induction method extended to the Marcus constituent buffer. Moreover, some of the same constraints that aid in local parsability -- roughly, the conditions that ensure that the grammar so defined will be LR(k) -- play a role in learnability as well. For instance, by restricting grammar rules to trigger over a finite parsing radius, we ensure that no single grammar rule will demand unbounded context for its proper operation. It seems then that the two functional demands of parsability and learnability are related. Chapter 4 discussed the trade-off between the two from an abstract standpoint. In this chapter we shall examine the same problem in the specific context of the Marcus parser. Just how do the LR(k) constraints affect learnability? How do these local parsability constraints relate to linguistic constraints presented in formalizations of theories of generative grammar? How are the constraints of the acquisition procedure related to those independently advanced by Wexler and Culicover [1980]? These questions are the topics of Chapter 5.

Sections 1 and 2 of this chapter formalize the Marcus parsing model. First, we show how packet names and node annotations can be replaced by complex state-stack symbol combinations. Second, we show that the derivation sequence traced out by a Marcus parser is right-most (using the results of Hammer [1974]). Third, we show how Marcus' "attention shift" mechanism is really just a version of Szymanski and Williams' [1976] *non-canonical* bottom-up parsing model. (This approach, in turn, is based on a suggestion of Knuth's [1965].) If only a finite number of attention-shifts are allowed, then Szymanski and Williams show that only the deterministic context-free languages can be handled.

In Section 2 we present a restricted 2-stack parsing model, designed to extend LR(k) techniques to non-context free languages. Using the finite automaton representation for parsers suggested by De Remer [1969], we present a necessary condition for a context-sensitive grammar to form the basis of an extended LR(k) parsing procedure: the set of *characteristic strings* must be a regular set. [1]

When applied to a context-sensitive grammar, this condition forbids rule systems where canonical

---

1. De Remer's definition of a characteristic string is as follows. Let {#0, #1,...,#s} be a set of symbols not in the terminal vocabulary of a grammar G, numbering the productions of G. Let the p-th production be $A \Rightarrow \omega$. Let there be a canonical derivation such that $S \overset{*}{\Rightarrow} \rho A \beta \Rightarrow \pi \omega \beta$. Then $\pi \omega \#p$ is a characteristic string of α.
As De Remer observes, the condition that the set of characteristic strings be regular is always satisfied by the right-most derivations of a context-free grammar, since one can write a set of right-linear productions for the set of characteristic strings: $S \dashrightarrow \omega_0 A_0$; $A_0 \dashrightarrow \omega_1 A_1$, etc.
A right-linear grammar generates only a regular set, of course.

derivations involve non-terminal center-embedded re-write sequences, i.e., successive sentential *forms* such that $\gamma A B \beta \overset{+}{\Rightarrow} \gamma D A C \beta$, where A, B, C, D are non-terminals. This is so because then the set of characteristic strings cannot be regular, by a well-known theorem of Chomsky [1959]. (Note here that we consider a set of characteristic strings, and not the language generated by a grammar.) This condition is plainly not *sufficient* to ensure deterministic parsing, however, since not all context-free grammars are LR(k). In addition to the representability condition, De Remer points out three other constraints required in order for a grammar to be LR(k): (i) its representation as a parsing automaton cannot have any so-called *inadequate* states, i.e., the transition diagram must be deterministic; (ii) the finite state automaton must halt on all inputs, i.e., it must not loop forever on some inputs; and (iii) it must detect errors in input sentences as early as is possible.

Viewed as an extension of the Marcus parser, the restricted two-stack model can handle some non-context free languages, e.g., $a^n b^n c^n$. To do this requires an arbitrarily long input buffer, however. We compare this result to the languages learnable in the Wexler and Culicover framework.

Finally, Section 3 of this chapter discusses the relationship between the locality principles prescribed by Lparsifal and those advanced in linguistic theory. In particular, it examines the Wexler and Culicover constraints on a mathematical model for the acquisition of a transformational grammar, and the Lasnik and Kupin formalization of a transformational grammar. In brief it will be shown that, the majority of the constraints advanced in both the Wexler and Culicover and Lasnik and Kupin models are shared by Lparsifal.

5.1 Formalizing the Marcus parser

Our first task is to that the Marcus parser is a variant of a conventional LR(k) parser, namely, a LR(k,t) parser (first proposed by Knuth [1965]).[2] There are two basic parts to this demonstration: (i) converting the Marcus transducer into a more typical parser; and (ii) showing that the Marcus parser in effect observes a right-most recognition order (neglecting the complication of attention shifts).

Here, the phrase "a more typical parser" means simply that there is a difference in the data structures used by the Marcus parser and parsers as they are usually presented in the literature. The basic difference is that the Marcus parser stores a partial representation of the parse tree on its stack, and makes reference to that representation to guide its parse; however, most formalized parsers do not store an *explicit* representation of the parse tree. For instance, an LR(k) parser stores complex

---

2. Specifically, it is a LR(2,2) parser: it uses at most two symbols of look-ahead and can reduce either one of two left-most complete sub-trees at any step. Below we will discuss exactly what this terminology means.

information on its push-down stack, essentially a table that acts like a miniature finite automaton dictating to what new state it should go and what action it should perform given such-and-such current input symbol, such-and-such look-ahead symbols, and its current state. The point of the table is to store all the possibilities of where, in possible right-most derivations, the parser could be. (Below we discuss this matter in some detail.) Intuitively, this is the work that the partial tree representation is doing in the Marcus parser. It should come as no surprise then that one could replace the *explicit* tree representation with an *implicit* state-symbol representation. Instead of building a tree, we simply number all right-hand sides of productions, corresponding to the Marcus grammar rules that were labelled "Attach as X." Each attachment made to a constituent in the active node stack corresponds to complete analysis of a phrase of type *X*; the modified parser announces this fact by emitting the number associated with the production. Similarly, suppose all other distinct rules are numbered. Then the output of a parse will simply be an (ordered) sequence of numbers, $p_1, p_2,...,p_n$. This is one conventional way of describing a parser's recognition sequence, as described in Aho and Ullman [1972 page 264]. Suppose we define a context-free grammar in the usual way, i.e., as a four-tuple of non-terminals (N), finite alphabet (Σ), productions (P), and start symbol (S). Then Aho and Ullman define a left (right) parse as follows:

> Let G = (N,Σ,P,S) be a context-free grammar, and suppose that the productions P are numbered 1,2,...,p. Let α be in $(N \cup \Sigma)^*$. [So α is just a sentential form.] Then:
>
> A *left parse* (*right parse*) of α is a sequence of productions used in a leftmost (right most) derivation of α from S.

As Aho and Ullman show, we can think of a parser for a grammar G formally as a simple transducer (a syntax directed translation system) that maps strings in the language of G (L(G)) to all their right or left parses. To actually build a device that does this work, one must *implement* this transducer, generally as a push-down machine of some kind. Successive operations of the transducer (and hence right- or leftmost derivations) correspond to successive configurations of this processor, in the familiar way.

Formally, let us define a push-down automaton (PDA) as a 7-tuple, $(Q, \Sigma, \Gamma, \delta\ q_0, Z_0, F)$. Q is a finite set of states, Σ a finite input alphabet, Γ a finite set of symbols that can appear on the push-down stack, δ a finite set of transitions or productions, q_0 an initial state, Z_0 the initial symbol on the push-down stack, F the set of final states. A *configuration* of the automaton is defined as (q, w, α) in $Q \times \Sigma^* \times \Gamma^*$. A move of the automaton, denoted $\vdash$, is a relation between two configurations; $(q, aw, Z\alpha) \vdash (q^*, w, \gamma\alpha)$ iff $\delta(q, a, Z) = (q^*, \gamma)$. To convert the PDA into a transducer, we need only add an output to the transitions γ to write out the appropriate production numbers; δ will then map from $Q \times (\Sigma \cup \{e\}) \times \Gamma$ to $Q \times \Gamma^* \times \Delta^*$. (Here, *e* is the empty string.)

How exactly can the partial subtrees used by Parsifal's grammar rules be encoded into the configurations of a PDT? Let us show informally how this is done. If the subtree consists of just a single symbol, then the answer is clear: the corresponding stack symbol of the PDA will just be the same. So Γ must include all the possible active node stack items -- NP, VP, S, PP, etc. Further, since an active node labelled, say, *+passive* is different from one not so labelled, we must distinguish between the two; we do this by forming complex symbols, e.g. [VP +passive], [VP]. There are only a finite number of such complex symbols, since there are only a finite number of such feature annotations. What if an active node is a partial sub-tree that has several daughters? As discussed at the end of Chapter 3, only the immediate daughters of an active node are assumed to be accessible to grammar rules. This means that if, say, an NP and a PP have been attached to a VP node that is being built, then these nodes are part of the VP sub-tree. However, none of the syntactic material *below* the NP or PP -- the N or its complements, for example -- can affect the parse, by assumption.[3] This means that the internal syntactic detail of an NP or PP node attached to some active active does not change the automaton's transitions. But this means, in turn, that one can encode the sub-tree attached to an active node as a single complex symbol: if the active node consists of a mother node X and daughters $Y_1,...,Y_n$, simply form the bracketed expression [X Y_1 ... Y_n]. If the number of possible daughters of a node is finite, the number of such complex symbols is finite, and the encoding is possible. Does this last condition hold? At first glance, it seems not to hold. For example, a VP can have an indefinite number of (adjunct) PP's attached to it: *I saw Bill behind the table beside the bed near my house....* To eliminate this possibility, we must be able to claim that such adjunct expressions can be finitely encoded as a single complex symbol, at least for the purposes of affecting the future course of a parse. Fortunately, in English this seems to be the case. A potentially infinite sequence of PP's, Adjectives, or other adjunct sequences X occur in the form $(X)^*$. Suppose we represent adjunct sequences of this kind as such. That is, suppose we form the complex symbol $[\ X\ ]^+$ whenever two or more such adjuncts appear. For example, the sequence PP PP PP PP attached to VP will be collapsed to [VP PP^+], indicating that some (non zero) number of adjunct PP's has been attached to a VP. Let us call this an adjunct cluster.

Note that the requirement that the items so represented be adjuncts means that the NP arguments to a Verb (alternatively, the NP's subcategorized by the Verb) will not be so collapsed. This implies that the number of "true" arguments to a phrase must be finite; e.g., that a Verb can have at most a finite number of arguments that are assigned thematic roles. In contrast, the adjunct cluster [PP] is assigned just a single thematic role. Given these assumptions, then the annotations on an active node are

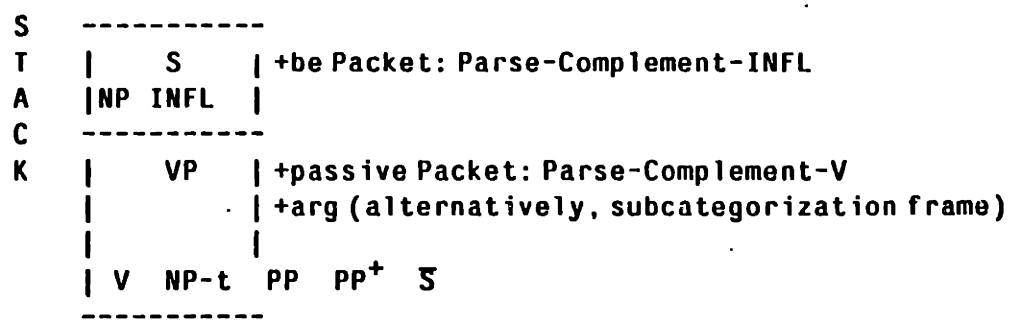
---

3. It is a matter for empirical confirmation as to whether this assumed constraint is descriptively adequate; it might be too strong, for example.

finite.[4] Therefore, we can encode any sub-tree relevant to parsing transitions as a single complex symbol.[5]

What about the packet names? Again one can just factor them into the complex "state" that is saved on the push-down stack. So for example, the packet Parse-Head-INFL will become simply the state q_i, the packet Parse-Complement-INFL the state q_j, etc. The actual stack symbol stored will be simply the pair [X, q], where X = the complex symbol described earlier and q = the state as just defined.

As an example of this encoding, consider the stack below, corresponding to the sentence, *John was believed by Fred with great delight... on Tuesdays to be a fool*.



This stack would have the encoding,

---

4. Observe that nothing rules out sequences of the form, adjunct cluster -- true argument -- adjunct cluster -- true argument, etc. But then, since the number of true arguments is finite, eventually one will obtain a finite sequence of alternations followed by some indefinitely long adjunct cluster -- still finitely representable as an annotation.

This method assumes some way to distinguish between true arguments and adjuncts. Some such method is required in any case, in order to determine whether, e.g., a Verb's subcategorization frame has been fulfilled.

5. So far, we have not said how binding may be carried out in such a model. For NP binding, matters are clear. Since an antecedent of an NP trace must satisfy Subjacency, the NP antecedent must be in the immediate cyclic domain in a non-thematically marked position. Therefore, it must be a daughter of the cyclic S, and is accessible according to the encoding already described.

Wh-traces are more complicated. Marcus posited a special *wh*-register, associated with each cyclic node, that could be either empty or filled with the name of a wh-constituent. There could be at most one wh-register per cyclic domain. Given this constraint, then the wh-annotation becomes simply a part of the complex symbol name associated with the appropriate S. The *wh* element is passed along by successive cyclic movement.
Some problems with this approach still remain, however. For example, there are the cases of long distance control, where an NP antecedent can appear arbitrarily far away from the element it binds: *They$_i$ know [that Bill believes [that [PRO]$_i$ feeding themselves would be difficult.*

This problem will be left unsettled here.

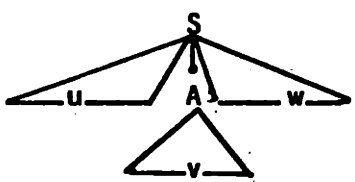
[(S NP INFL be], q_i]
[(VP passive ary NP-t PP PP^+ S), q_j]

where q_i and q_j denote states corresponding to the appropriate packets.

At the beginning of a parse, the stack will contain the special start symbol S^*. Acceptance will be by empty stack plus the symbols S^* \$ in the first two cells of the input buffer; the machine enters an *accept* state and halts. An error condition is signalled if, at any point in the parse, no actions can be taken and the machine is not in the accepting state. (Thus this machine is *not* a version of the acquisition system, but rather a "mature" parser.)

Finally, since grammar rules can make reference to not only the top element of the stack, but also to the so-called *current cyclic node*, we must have some way to encode this information as well. When a new active node is formed, its current cyclic node is simply the cyclic node of the node that was just active, unless a new cyclic (NP or S) node has been formed in the meantime. In either case, we can simply code an active node as the pair [active node, cyclic node]. Since the annotations on any cyclic node are finite, this representation is finite as well.

The formation of complex symbols in this way is in fact intimately related to the standard development of LR(k) parsing. Let us review this material here, since it will be important in the sequel. First we consider the following intuitive definition of an LR(k) grammar. "We say that [a grammar] G is LR(k) if when examining a parse tree for G, we know which production is used at any interior node after seeing the frontier to the left of that node, what is derived from that node, and the next *k* terminal symbols." (Aho and Ullman [1972, page 379]). The picture of a derivation step looks schematically like this:



According to the LR(k) condition, if G is LR(k), then we can decide deterministically to reduce *v* back to A, given that we have already built the parse tree corresponding to *u* and can look at the first *k* symbols of *w*. Formally, we may say that a context-free grammar is LR(k) if,

(1) $S \overset{*}{\Rightarrow} \alpha A w \Rightarrow \alpha\beta w$,
(2) $S \overset{*}{\Rightarrow} \gamma B x \Rightarrow \alpha\beta y$, and

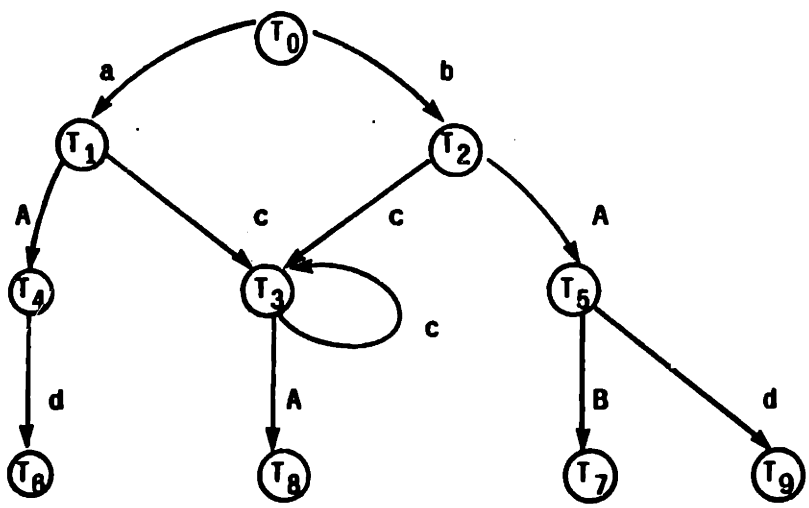
(3) the first k symbols derivable from w = the first k symbols derivable from y
imply that
$\alpha = \gamma$, $A = B$, and $x = y$, i.e., that only one reduction is possible.

(The derivations in this case are rightmost.)

One problem here is that the left-hand context looks like it could be infinitely long. As Aho and Ullman note, "the LR(K) condition says that we can uniquely determine the handle of a right-sentential form if we known all of the right-sentential form in front of the handle as well as the next k input symbols." (*Ibid.*, page 380) However, it can be shown that if a grammar is LR(k) then a finite table associated with any top-most stack element is sufficient to uniquely determine what to do next. This result makes use of the fact, cited above, that the sentential forms of a rightmost derivation of *any* context free grammar (and some non-context free grammars) form a regular set. Following DeRemer [1969], this property allows one to represent the finite control table information required by an LR(k) parser as a finite automaton. Consider an example of this provided by Harrison [1978 page 526]:

Grammar:
$S \Rightarrow aAd$ $S \Rightarrow bAB$
$A \Rightarrow cA$ $A \Rightarrow c$
$B \Rightarrow d$

As may be verified, the set of right-sentential forms with respect to this grammar is: {aAd, acc^*A, bAd, bAB, bcc^*A}. This set is regular. Therefore, we can represent it via the following simple transition network (Harrison [1978 page 528]), where the T_i are states to be defined later.



Each path in the transition diagram terminates in the completion of a particular constituent. These terminal states are called *reduction states*, because they signal the completion of constituents according to the rules of the grammar. T_6 announces the completion of an S, since it corresponds to the sentential form *aAd*; likewise, T_8 corresponds to the completion of an A, T_7 to an S (*bAB*) and T_9 to a B (*d*).

This control information is not enough to carry out a parse, of course, because a reverse derivation consists of a *sequence* of right-sentential forms. Hence in general we must make several "passes" through the network. (This description is taken from DeRemer [1969].) For instance, the sentence, *bccd* is rightmost-derived via the productions, S⇒bAB⇒bAd⇒bcAd⇒bccd. Hence, in reverse, we recognize this sentence by recognizing the sentential form *bccd*. The parser works by scanning the string left-to-right. It first must recognize the sequence *bcc* as the sentential form *bA*. This it can do: a *b* followed by any number of *c*'s leads to state T_8, with an A recognized. It then can complete the parse because the form *bAd* will force a transition to recognize *d* as a B, and then, finally, *bAB* completes the construction of an S. Note that the number of passes through the network corresponds the number of steps in the derivation, but that the passes were carried out in reverse order, as expected. Evidently, the control network is called recursively, and this requires a stack. To take the example just described, while the parser was analyzing the form *bcc...*, it had to analyze the fragment *cc* as an A, and then returned to consider the form *bA...*. But this sequence of A's could be indefinitely long. To remember where in the analysis it was interrupted in general requires a stack, with each stack symbol storing enough information about the parse to restore the environment at the time any recursively constructed node are completed. In this case, it is the "states" T_i, also called *LR(k) tables*, that are pushed and popped on a stack.

In more detail, we can follow Harrison [1978] and define an LR(k) parser as follows. First we define the transitions of the control network, adding the actions to either (i) shift input items and state names onto the push-down stack; (ii) pop some series of input items and state names (indicating a reduction); or (iii) indicate an error condition.

> Let G=(N, Σ, P, S) be a (reduced) context-free grammar and let $k \geq 0$. Let $\mathcal{T}$ be a set of states. Define two functions *f* and *g* as follows.
>
> 1. *f*, the *parsing action function*, is a mapping from $\mathcal{T}$ x {x∈ into {shift, error} ∪ {reduce π|π∈P}.
> 2. *g*, the *goto function*, is map from $\mathcal{T}$ x N into $\mathcal{T}$ x {error}.
>
> (from Harrison [1978 page 527])

Thus *f* looks at the current "state" as indicated by the current stack symbol $T_i \in \mathcal{T}$ and the current input symbol(s), and decides whether to push the symbol (along with the associated T_i) onto the stack, pop the stack (reduce), or declare an error. It thus corresponds to the grammar rule actions of the Marcus parser; we shall describe exactly how in more detail later. The function *g* is the transition function of

the finite state network representation of the parser. It indicates to which state the parser should go, given some current stack symbol T_i and input symbol(s), and declares an error if no move can be made.

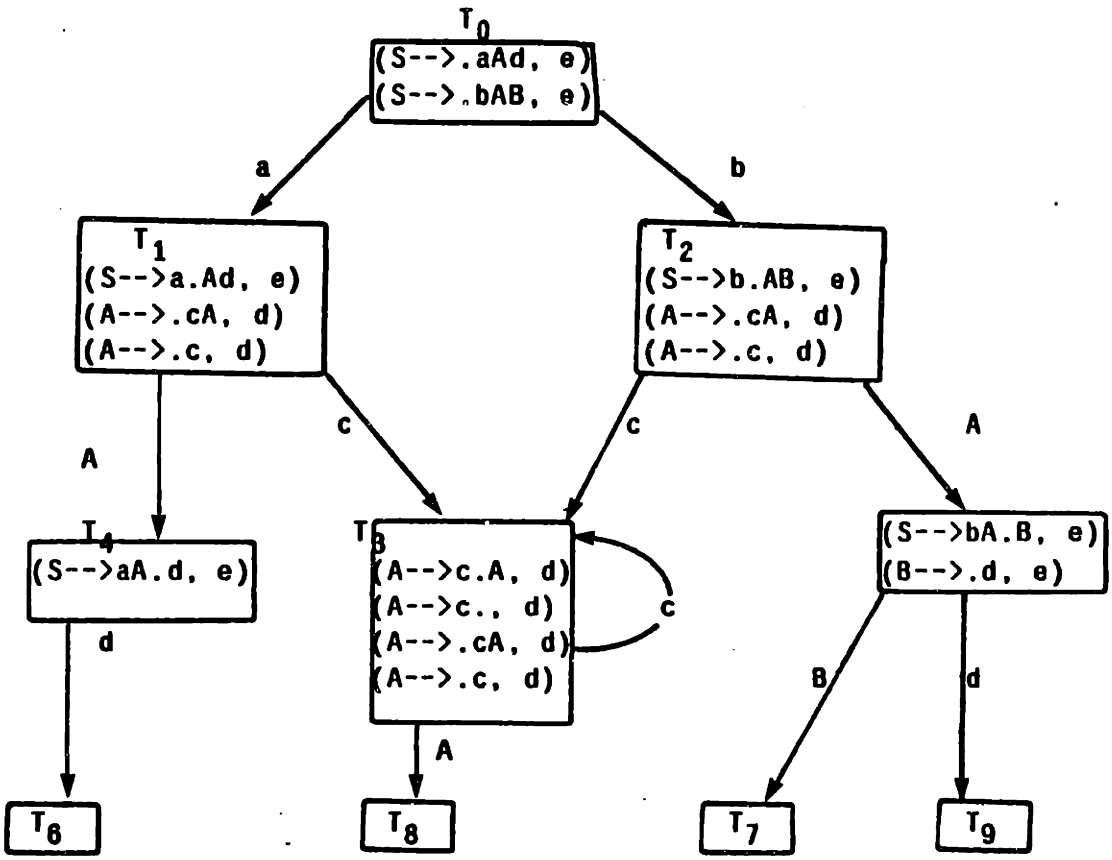
What information is contained in these tables? In brief, each T_i holds a (finite) representation of where in the analysis of a right-sentential form the parser could be, given that *k* tokens of look-ahead information is available. For example, after seeing a *b* then the parser must be in one of the following states: (1) it could have just finished the *b* portion of the rule $S \Rightarrow bAB$, with A and B completely built or A built and *d* to check; (2) it could have seen *b* and already have built the last A in a string of A's, with B or just *d* to check; or (3) it could have seen *b* and not yet built an A (or possibly many A's). These possibilities are called the *set of valid LR(k) items*, for this grammar, production, and look-ahead. An LR(k) item is defined with respect to a grammar G as the pair, $(A \Rightarrow \beta_1 . \beta_1, w)$, where $A \Rightarrow \beta_1\beta_2$ is a production, and *w* is a look-ahead of at most *k* terminal items that can be derived from A. (Intuitively, as observed by Harrison [1978], we have already checked that the derivation up to the dot is possible, and the look-ahead is *w*.) An item $(A \Rightarrow \beta_1 . \beta_2, u)$ is *valid* if there exists a derivation in G such that $S \overset{*}{\Rightarrow}_R \alpha A w \Rightarrow_R \alpha\beta_1\beta_2$, with u a valid look-ahead. The computation of the set of valid items, and the proof that they define the states of parser in the right way so that the resulting machine works correctly is the key to the LR(k) parsing method. For proofs, see Aho and Ullman [1972]; they will be omitted here.

Note that there are no conflicts in the resulting machine as to which transition to make. Specifically, there is never a conflict between a reduce or shift move, nor is there ever a choice of which reduction to make. If this is the case, we say that the resulting set of LR(k) items (a state) is adequate, otherwise, inadequate. An inadequate state has a conflict between whether to enter a final state, thereby reducing a set of items to some non-terminal, or to keep going with the analysis of the phrase.[6]

It is the sets of LR(k) items that form the tables T_i. The ten sets that form the states of the finite control required by the grammar above are as follows. (From Harrison [1978 page 535]. *e* denotes the empty string.)

---

6. More formally, as defined by De Remer [1969 page 47]: Any state having two or more transitions, at least one of which is a transition to a final state, is called an inadequate state.



For our purposes, it is sufficient to point out that the dot is used to mark the place of where in the derivation checking process we are, and this corresponds to the use of packets in the Marcus parser. For example, consider the expansion, NP⇒Determiner Adjective N Complement. If the packet Noun is active, then this means that we have already traversed the Determiner and Adjective portions of the rule -- i.e., that we have gotten as far as the N. But this means that the "dot" should be located as follows: NP⇒Det Adj $_{\uparrow}$ N... Packet deactivation and subsequent packet activation corresponds to moving the dot, since this means that we have completed the construction of a constituent or determined that some constituent is optional. Moving the dot leads to a new state of the finite control network, as indicated above, with a new group of LR(k) items active. The set of LR(k) items is just the set of possible rule expansions that are possible at a given point in the parse. We see then that a packet of grammar rules, in Marcus' sense, corresponds simply to a set of LR(k) items. The stack of active nodes and packets plus buffer contents of the Marcus parser correspond precisely to the stack of appropriate dotted rules plus look-ahead items (valid LR(k) items). Neglecting for the moment the apparent use of top-down predictions in the Marcus parser and its use of non-terminal look-ahead, we can summarize the discussion so far by saying that the Marcus parser is just an LR(k) parser.

Let us now consider these last two issues in turn. First of all, it seems as if the Marcus parser could not be LR(k)-based because, as Marcus himself claims, sometimes it makes top-down *predictions* rather than working in a strictly bottom-up fashion. For example, if the item *the* is seen in the input buffer, then a Noun Phrase is predicted and placed on the active node stack, without ever waiting for the remainder of the right-hand side elements that confirm the existence of an NP. Of course, Parsifal then proceeds to *check* that in fact an NP can be found; since the system is assumed to be deterministic, though, this check is presumably guaranteed unless the sentence is ill-formed. (For example, a non-sentence could be, *the gave the book.*)

As pointed out by Hammer [1974], however, one can design a bottom-up parser that integrates top-down prediction in the same fashion as Parsifal has. Let us quote from Hammer here:

> Prediction is a concept usually associated with top-down parsing, rather than bottom-up. The entire process of top-down parsing consists of nothing more than making a sequence of predictions and then seeing them come true. Given a nonterminal and a lookahead string, we predict which rule of the grammar must be applied to the nonterminal in order to effect the eventual generation of the lookahead string...
>
> On the surface, this concept seems completely alien to the approach embodied in bottom-up parsing. There, nothing whatsoever is predicted, no anticipation of the future ever plays a role in the progress of a parse. It is only when we have reached the end of the handle that we recognize it as such and discover to which nonterminal it is to be reduced. ... But in one important sense, there is an air of prediction to the entire proceedings. Before the bottom-up parse of a sentence begins, we are effectively predicting that we shall reduce it to S, the sentence symbol. ... We will generalize this approach and utilize prediction not only at the beginning of a bottom-up parse, but at selected points throughout. That is, at various times we shall predict that the bottom-up parser, proceeding in a normal fashion, will eventually reduce some prefix of the remaining input to a specified nonterminal.
> [pages 27-28]

Hammer goes on to show that we can regard this inter-weaving of top-down prediction with bottom-up reductions as in effect a set of possible subroutine calls. At certain points in the parse, the "main" parsing machine can hand over the job of finding a particular, predicted nonterminal to a subroutine. Again to quote Hammer:

> ...It is easier to think of [the new machine]... as representing a generalized kind of parsing machine, in which it is possible for one state to transfer control to another part of the machine and get it back when that part has completed its predefined task. Each time a call transition is made, a new stack is created for the benefit of the called submachine. This submachine will use this new stack as its parsing stack, and will destroy it when control is returned to the calling state. Since (as we shall see) it is possible for submachine calls to be nested (i.e., for one submachine to call another, or itself recursively), it is more convenient to think of the machine at large as really having a stack of stacks...
>
> It should be observed that the order of recognition by such a machine is still strictly

left-to-right, bottom-up. All the extra work of making predictions and fulfilling them, of creating and suspending stack levels, as of yet makes no appreciable difference. A comparison is made in Figure 3.4 [reproduced below as Figure 5.1] between a conventional LR parse and one utilizing this predictive capability. Actually, it is a portion of a parse, showing how the substring $xy_7y_5y_3y_1$ would be handled... assuming that the x takes us from some state q_0 into state q_1. (We do not know or care what is on the stack underneath the q_0, since it is unaffected by this segment of the parse.)

In the second part of the figure, each stack level is written horizontally; laying the stack levels end to end approximately reproduces the single stack of the first parse. In the revised parse, an E is predicted upon entry to the state q_1; this causes an E to be written on the old stack level, and a new stack level to be created, which uses state q_{15} as its starting state. Thereafter parsing continues normally until the POP state is entered [i.e., the E is confirmed]; then the top stack level is destroyed, and the machine resumes processing on the lower level, transferring to the E successor of the state from which the call had been made. Note that there are two more steps in the predictive parse, to account for the prediction and fulfillment steps, but otherwise the parses and stack configurations are essentially the same. **In particular, the final states are entered in precisely the same order, thus preserving the order of recognition in the parse.** [my emphasis, rcb]
[*Ibid.*, pages 37-39]

Split LR machine (from Hammer [1980, page 36])

(Final states are circled.)

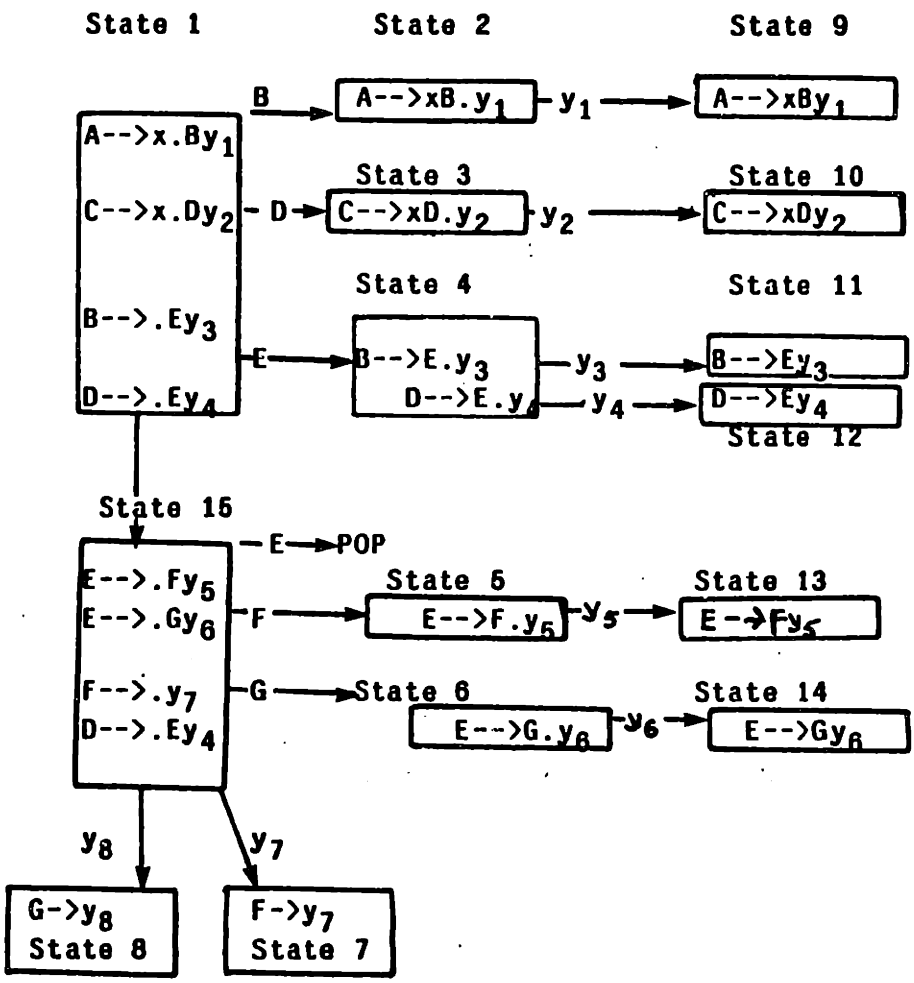
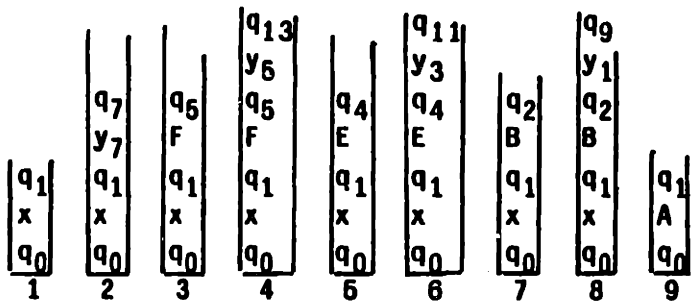


Figure 5.1(a) - Split LR(0) machine

Conventional LR(0) parse



Revised parse with prediction

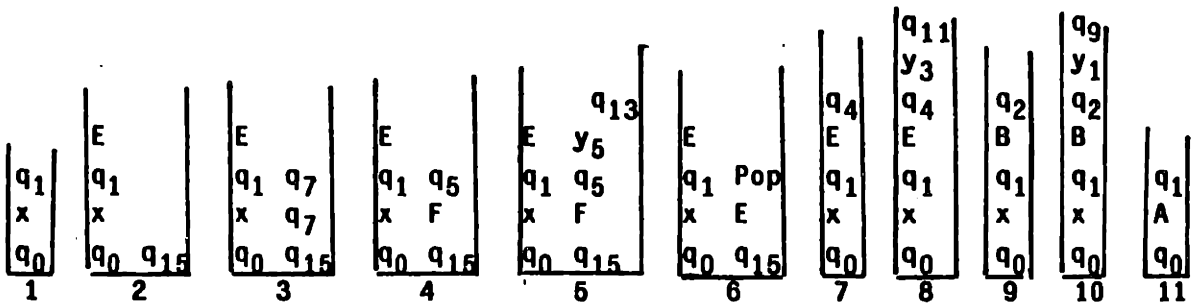


Figure 5.1(b) - Comparison of conventional and subroutine parses.

Recall that sets of items correspond to packets. The exit from a set of LR(k) items (= a state in the diagram) corresponds to the deactivation of a packet, and entry into a set of items corresponds to the activation of a packet. Completion of a phrase (=entry into a final or reduce state of the control automaton) corresponds to dropping a phrase into the input buffer in the Marcus machine. It should be apparent that, given this correspondence, the use of top-down prediction in Parsifal is subsumed by the Hammer formalization. Attachment of a node to an active node occurs in precisely the order specified by Hammer, that is, in a bottom-up, left-to-right order, even though we may *predict* the existence of a node and place it in the active node stack before it is completely recognized.[7]

Actually, Marcus only explicitly allowed this "stack of stacks" approach in the analysis of Noun Phrases. To take a simple example, consider how NP's are parsed by the Marcus machine. A "leading edge" of an NP, say a Determiner, is recognized in the input buffer as prompting entry into the packet system for NP's. Marcus recognized this as a kind of subroutine call, what he dubbed an "attention shift." If an attention shift occurred, then the current parse would be suspended, and the parser entered recursively to handle the NP. Parsifal's packet for NP's corresponds precisely to Hammer's notion of a submachine -- the parser is called recursively. When an NP has been successfully parsed, it is returned to the input buffer, and the parse picks up where it left off. If the attention-shift occurs with a triggering item in the first buffer cell, then this is exactly Hammer's approach. For now we ignore the fact that the NP could be recognized non-canonically. As we will see, this occurs if an "attention-shift" is allowed to occur on an item in the second or third buffer cells.

Besides NP attention shifts, there are other places in Parsifal were top-down prediction occurs. These cases correspond to what we earlier called the "automatic" creation of nodes, for example, the creation of a PP node given that a Preposition has been detected in the input buffer. These node creations plainly fall under Hammer's design as well. Since NP attention shifts and node creation via grammar rules (now handled by detection of Heads) are the only places where top-down predictions are used in Parsifal, we can conclude that Hammer's formalization subsumes Marcus' use of top-down prediction, and that in fact the recognition order of phrases is strictly bottom-up.[8]

Finally, let us consider the possibility of non-canonical recognition sequences. Chapter 1 discussed a case where one had to delay the recognition of an item in the input until a complete constituent to the

---

7. This statement is not quite accurate, in that there are a few places where Marcus machine predicts a node *and* attaches it to its proper place in the parse tree. In particular, VP nodes are created and attached to the S's that dominate them without first completing the construction of the VP. This cuts against the grain of the rest of the parser, however, since the VP node still remains in the active node stack. If uniformity and a strict bottom-up recognition order is to be maintained, then we must change this so that nodes in the active node stack cannot be attached to any mother nodes. The only way for attachment to a superior to occur is to (1) complete the construction of the node then (2) drop the node into the buffer and finally (3) attach the completed node. Direct attachment would complicate the acquisition procedure.
8. Save for the use of second- or third-cell attention shifting and the caveats discussed in the preceding footnote.

right of that item was completely recognized. The example was a situation where *Have* could be either an Auxiliary verb or a Main verb: *Have the boys take/taken the exam.* The parser could decide which only after parsing *the boys...* as a complete NP; then it could look at the look-ahead item *take/taken.* In this case then we have delayed deciding about the rules Auxverb-->have or Main-verb-->Have until the Subject NP has been recognized. Therefore, this recognition sequence is non-canonical -- a reverse rightmost (or leftmost) recognition sequence is not followed. Nevertheless, the parse is still bottom-up.

Szymanski and William [1976] have studied parsing models of this kind. The approach is based on a suggestion of Knuth's [1965] called LR(k,t) parsing. In a standard LR(k) parser, one must always attempt to reduce the left-most complete sub-tree. For example, in the case of *Have the boys...*, *Have* is the left-most complete sub-tree, and therefore one is limited to deciding what to do with *have* before one proceeds to analyze *the boys...*. However, in LR(k,t) parsing, one is allowed to reduce any of the t left-most sub-trees. For example, an LR(2,2) parser can handle the *Have the boys...* case, because it is permitted to reduce the NP starting with *the boys...*, the second left-most complete sub-tree, and then use a look-ahead of two items, NP and *takes/taken*, to decide what to do next. To quote them,

> Parsers for LR(k,t) grammars can be constructed using a technique similar to that used for building LR(k) parsers...The only major difference is that the lookahead string associated with an individual LR(k) item is allowed to contain nonterminals as well as terminals. Whenever an inadequate item set (i.e., one in which the correct parsing action is undefined due to the presence of conflicting items) is reached, the parser postpones any implied reduction(s) and shifts to a new set of items. This new item set includes in its closure any items produced by expanding the leading nonterminal in the lookahead string of any postponed items...
>
> Since the process of postponing an individual item can be repeated at mosdt t times, no lookahead string need have a length that exceeds kt. Hence the above construction is finite and we have the next theorem.
>
> Theorem 4.2 *It is decidable whether an arbitrary CFG is LR(k,t) for fixed values of k and t. Furthermore, an LR(k,t) grammar can be parsed in linear time.*
> [1976, page 241]

Moreover, Szymanski and Williams observe that LR(k,t) grammars can be parsed by a deterministic push-down automaton, since one need back up at most a finite number of times in succession, over a finite distance. Therefore, LR(k,t) languages are deterministic languages. (For every LR(k,t) language, there must be some weakly equivalent LR(k) language, by the equivalence of LR(k) with deterministic languages.)

How does this result fit in with Marcus' notion of attention-shifts? Note that the LR(k,t) restriction says that only a *finite* number of attention shifts should be allowed. This is in fact what Marcus proposed; he limited the attention shifts to three. What happens if we allow an unlimited number of

attention shifts? Szymanski and Williams show that then some (unambiguous) non-deterministic languages become parsable. For example, consider the non-LR language,

$$L=\{a^n b^n c^m d^{m-l} | n,m,l \geq 1\} \cup \{a^n b^{2n} c^m d^m | n,m \geq 1\}$$

As Szymanski and Williams observe, this language cannot have an associated LR(k) grammar, for any *k*, because "any grammar for [L] will require the handle of some sufficiently long sentence to be at the *a-b* interface, and the context required to distinguish between two alternative parses, namely, whether there are more *d*'s than *c*'s in the remaining portion of the string, cannot be determined by a partition into regular sets." [1976 page 232] But there is a non-canonical deterministic parser for this language: one need merely postpone the decisions about the *a*'s and *b*'s until we find out about the *d*'s and *c*'s. This is precisely what happens in the *Have the boys...* case. Note that the analysis of L may in general demand an unbounded number of postponements, or "attention shifts." Szymanski and Williams go on to show that if one limits the parser to an unbounded number of postponements, restricted to the case of left- and right- contexts that are representable as regular sets, then the resulting parser still operates in linear time on a two-stack deterministic push-down automaton.[9]

If the Marcus parser fits into Szymanski and Williams' model as described, then it is actually an LR(2,2) parser: at most one non-nested attention shift is allowed, and two look-ahead symbols are used. A single non-nested attention shift translates into the ability to reduce one of two left-most complete sub-trees, at any step. Why do only two symbols count for look-ahead, when the Parsifal input buffer can hold three elements? The first item in the input buffer serves as a locus for reductions; it is actually an input token that prompts control transitions, rather than a look-ahead symbol *per se*. For example, even an LR(0) parser, one that uses *no* lookahead, must read input symbols from time to time; these would fill the first buffer position. Therefore, it is more accurate to say that Parsifal uses two look-ahead tokens, and one input token.[10]

Let us summarize the analysis so far. Parsifal is essentially an LR(2,2) parser Its use of top-down predictions can be folded into its bottom-up operation along the lines suggested by Hammer [1974], adding at most a linear number of steps to a parse. Packets and associated grammar rules correspond to sets of LR(k) items, as conventionally described. Parsifal's attention shifting mechanism for parsing NP's amounts to the use of non-canonical parsing techniques in a limited context. A finite amount of attention shifting does not add to the class of languages that can be parsed, namely, the deterministic context-free languages; it could add to the transparency or naturalness of the grammars that could be

---

9. If general left and right contexts are allowed, or if non-context free grammars are used, then n^2 time may be required. We give an example in the next section.

10. Incidentally, it is a well-known result (see Aho and Ullman [1972]) that every deterministic language has an associated LR(1) parser, though not necessarily a very transparent one. This makes claims about the number of buffer cells required to successfully parse some subset of English difficult to evaluate.

written for those languages, just as it is often the case that a language is better expressed as an LR(3) rather than an LR(1) grammar, even though all deterministic context-free languages have LR(1) grammars. Finally, the parser so described executes in time linearly proportional to the length of input sentences.

5.2 Two-stack parsing

In this section we will define a two-stack parsing model that mimics the stack and buffer system of Parsifal. The machine is defined to have the early error-detection capabilities of an LR(k) machine. We will see that if an extension to Parsifal is made such that the input buffer "stack" can be arbtrarily long, then certain non-context free languages become parsable, in time n^2.

First, we observe that Parsifal has in effect two stacks: (1) its active node stack; and (2) its input buffer. The input buffer functions as a stack because items are pushed from the active node stack onto the input buffer when an item is *dropped* into the buffer, and popped from the buffer onto the active node stack when a completed constituent is *attached* to some active node. Crucially, the input buffer acts as a stack only if we assume that what Marcus called the "left-to-right" hypothesis holds, namely, that items are attached from the input buffer to the active node stack in a strict left-to-right order.

We can picture this machine as follows:

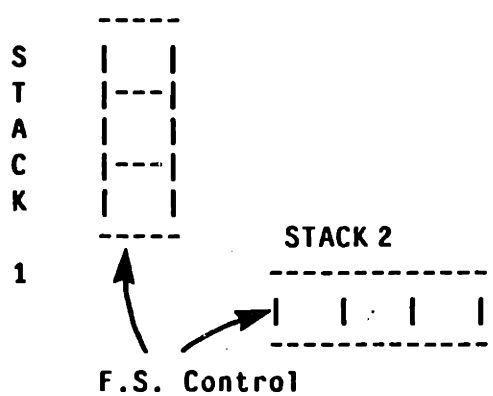


Figure 5.2 - The Marcus parser as a 2-stack parser.

A formalization of this machine can proceed along standard lines. Instead of just a single set of stack symbols Γ, we include a separate set of symbols for the active node stack and for the input stack. Active node stack symbols are of the form, (X,q), where X = a non-terminal that can appear in the

input stack, and q = a state of the finite state control. A configuration of the machine at step *i* (a snapshot of its state) can be written by concatenating the two stacks together along with the state at step *i*; this follows Walters [1970] notation for his extension of LR(k) parsing to a 2-stack machine. Thus a configuration looks like, [Stack 1 · state · Stack 2]. The instructions of the machine can be either (1) a reduction, in which case the active node stack is popped and the completed non-terminal pushed onto the first cell of the buffer, and the state is possibly changed; (2) a look-ahead transition, in which case the state of the parser may change without altering any items in the input buffer or node stack; or (3) a read transition, possibly using look-ahead, in which case an symbol from the input stack may be placed on the active node stack, and a state transition can occur. In addition, we must define an accepting configuration, in the usual way. Note that this parser can use non-terminal look-ahead symbols, since non-terminals may be dropped into the input stack, only to be used as look-ahead material.

To see how this device works in practice, consider how we might build a machine to handle the non-context free language $a^n b^n c^n$. A suitable context-sensitive grammar for this language is the following. (We will discuss shortly a necessary condition on grammars for this construction to work.) As usual, the productions are numbered, #0, #1, etc.

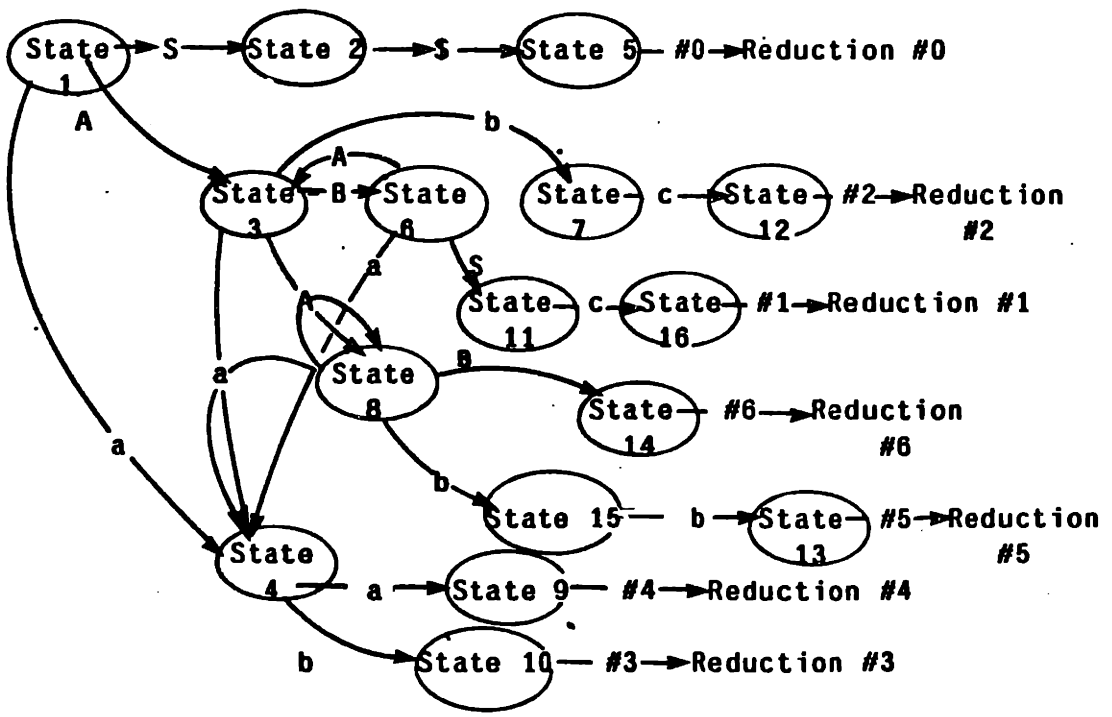
```
Start⇒S$ #0      Aa⇒aa #4
S⇒ABSc #1        Bb⇒bb #5
S⇒Abc #2         BA⇒AB #6
Ab⇒ab
```

We still the analogue of a "rightmost" derivation for context-sensitive grammars. Although the problem here is complex, the following notion will suffice here: following Walters [1970] define a derivation as rightmost if the successive lines of a context-sensitive derivation are formed without ever performing a re-write that lies wholly to left of an immediately preceding re-write. For example, the following derivation is "rightmost" in this sense, where the vertical bars indicate the material being re-written at the current step, and then angle brackets indicate the output material from the *previous* step:

```
 Start
   |S|
<AB|S|c>
A|B<A|bc>c
A<A|B>b|cc
A|A<b|b>cc
|A<a|b>bcc
<aa>bbcc
```

Note that at no point is a substring re-written that is entirely to the left of the portion just re-written in the previous line. As usual, the parse of this sentence will be in the reverse order of its generation; *aa* will be reduced back to Aa, etc.

We can now define a finite state control to recognize the right-sentential forms generated with respect to this grammar. The following is from Turnbull [1975]:



Let us see how a parsing machine of the sort defined above would handle $a^3b^3c^3$, as generated by this grammar. It takes 41 moves in all. The successive configurations (node stack -- state -- input stack) are as follows:

Step 1. $\$ q_1$ aaabbbccc$

Action: read a, go to state 4.

Step 2. $ [a, 1] 4 aabbbccc$

Step 3. $ [a, 1][a, 4] 9 abbbccc$
Read a, go to state 9.

Step 4. $ 1 Aaabbbccc$
Reduce via production #4.

Step 5. $ [A, 1]3 aabbbccc$
Read A and go to state 3.

Step 6. $ [A, 1][a, 3] 4 abbbccc$
Read a, go to state 4.

Step 7. $ [A, 1][a, 3][a, 4] 9 bbbccc$
Read a, go to state 9.

Step 8. $ [A, 1] 3 Aabbbccc$
Reduce via production #4

Step 9. $ [A, 1][A, 3] 8 abbbccc$
Read A, go to state 8.

Step 10. $ [A, 1][A, 3][a, 8] 4 bbbccc$
Read a, go to state 8.

Step 11. $ [A, 1][A, 3][a, 8][b, 4] 10 bbccc$
Read b, go to state 10.

Step 12. $ [A, 1][A, 3]8 Abbbbccc$
Reduce via production #3.

Step 13. $ [A, 1][A, 3][A, 8] 8 bbbccc$
Read A and stay in state 8.

Step 14. $ [A, 1][A, 3][A, 8][b, 8] 15 bbccc$
Read b, go to state 15.

Step 15. $ [A, 1][A, 3][A, 8][b, 8][b, 15] 13 bccc$
Read b, go to state 13.

Step 16. $ [A, 1][A, 3][A, 8]8 Bbbccc$
Reduce via production #5

Step 17. $ [A, 1][A, 3][A, 8][B, 8] 14 bbccc$
Read B, go to state 14.

Step 18. $ [A, 1][A, 3]8 BAbbccc$
Reduce via production #6.

Step 19. $ [A, 1][A, 3][B, 8] 14 Abbccc$
Read B, go to 14.

Step 20. $ [A, 1] 3 BAAbbccc$
Reduce via production #6.

Step 21. $ [A, 1][B, 3] 6 AAbbccc$
Read B, go to state 6.

Step 22. $ [A, 1][B, 3][A, 6] 3 Abbccc$
Read A, go to state 3.

Step 23. $ [A, 1][B, 3][A, 6][A, 3] 8 bbccc$
Read A, go to state 8.

Step 24. $ [A, 1][B, 3][A, 6][A, 3][b, 8] 15 bccc$
Read b, go to state 8.

Step 25. $ [A, 1][B, 3][A, 6][A, 3][b, 8][b, 15] 13 ccc$
Read b, go to state 13.

Step 26. $ [A, 1][B, 3][A, 6][A, 3] 8 Bbccc$
Reduce via production #5

Step 27. $ [A, 1][B, 3][A, 6][A, 3][B, 8] 14 bccc$
Read B, go to state 14.

Step 28. $ [A, 1][B, 3][A, 6] 3 BAbccc$
Reduce via production #6.

Step 29. $ [A, 1][B, 3][A, 6][B, 3] 6 Abccc$
Read B, go to state 6.

Step 30. $ [A, 1][B, 3][A, 6][B, 3][A, 6] 3 bccc$
Read A, go to state 3.

Step 31. $ [A, 1][B, 3][A, 6][B, 3][A, 6][b, 3] 7 ccc$
Read b, go to state 3.

Step 32. $ [A, 1][B, 3][A, 6][B, 3][A, 6][b, 3][c, 7] 12 cc$

Read c, go to state 12.

Step 33. $ [A, 1][B, 3][A, 6][B, 3] 6 Scc$
Reduce via production #2.

Step 34. $ [A, 1][B, 3][A, 6][B, 3][S, 6] 11 cc$
Read S, go to state 11.

Step 35. $ [A, 1][B, 3][A, 6][B, 3][S, 6][c, 11] 16 c$
Read c, go to state 16.

Step 36. $ [A, 1][B, 3] 6 Sc$
Reduce via production #1.

Step 37. $ [A, 1][B, 3][S, 6] 11 c$
Read S, go to state 11.

Step 38. $ [A, 1][B, 3][c, 11] 16 $
Read c, go to state 16.

Step 39. $ 1 S $
Reduce via production #1.

Step 40. $ [S, 1] 2 $
Read S, go to state 2.

Step 41. $ Start $
Reduce via production #0

Note that in order to perform the interchange of A's and B's that the input buffer must be arbitrarily large. This example cannot be handled by an unmodified Parsifal. It is also easy to show that the analysis will take at most n^2 steps, where n = the length of the input sentence.

We should also point out an important condition on sentential forms that permits this finite state control model to work. This is that the set of right-sentential forms (now in the extended sense of "right") must be a regular set. This is plainly a necessary and sufficient condition to be able to write down the required control network as a finite state automaton. Some strictly context-sensitive grammars do not have right-derivations that form regular sets, in contrast to all context-free grammars. For example, consider the following grammar for $a^n b^n c^n$ from Turnbull [1975]:

```
S⇒aSBc   bC⇒bc
S⇒aBC    cC⇒cc
aB⇒ab    CB⇒BC
bB⇒bb
```

The right-sentential forms of this grammar are not regular, since they include strings of the form $a^n b^m bc$, with $n \geq m+1$. If one compares the successive lines of a rightmost derivation in this case versus the preceding grammar, one can see just why this happens; the successive lines are self-embedding, that is, include non-terminal material to both the left and the right. The vertical bars in the figure below mark the intervals of re-writing for a given step, and the angle brackets represent the output of the re-writing rule from the previous step.

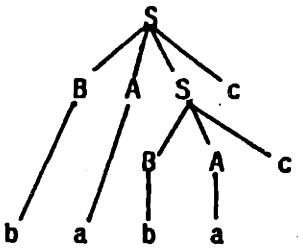
```
  Grammar 2          Grammar 1
(self-embedding)   (non-self-embedding)

    Start              Start
   <a|S|Bc>          <AB|S|c>
  a<|aB|C>Bc         A|B<A|bc>c
  a<ab>|CB|c         A<A|B>b|cc
  aa|b<B|C>c         A|A<b|b>cc
  aa<bb>|Cc|         |A<a|b>bcc
  aabb<cc>           <aa>bbcc
```

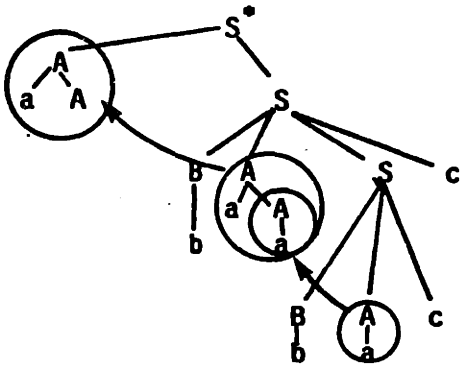
Observe that Grammar 1 never derives non-terminal material to the right of successive lines marked by the right-most vertical bar, whereas Grammar 2 does. By a well-known theorem (Chomsky [1959]), this means that the derivation lines of Grammar 2 cannot be represented by a finite-state machine, as required for our parser control. This is then a necessary condition for a deterministic parser of the kind used in LR(k) parsing theory, because otherwise one cannot even represent the control needed via a finite automaton. Importantly, even some strictly context-sensitive grammars are amenable to LR(k)-like parsing, as we have seen. Thus we see again that context-freeness, in itself, is not directly relevant to the issue of deterministic parsing.

As mentioned earlier, this restriction is plainly not a sufficient condition for deterministic parsing, since there are context-free grammars that are not LR(k) -- for example, ambiguous grammars. We must add other conditions to ensure that there will be no parsing conflicts. When we do this for the 2-stack parser presented above, we find that we must simply *define* that the machine will (1) detect errors as soon as possible, that is, on the smallest prefix x such that xy cannot be in L(G), for all possible y's consistent with the look-ahead seen so far; and (2) not loop indefinitely. This development is covered in Turnbull [1975] and will not be discussed here.

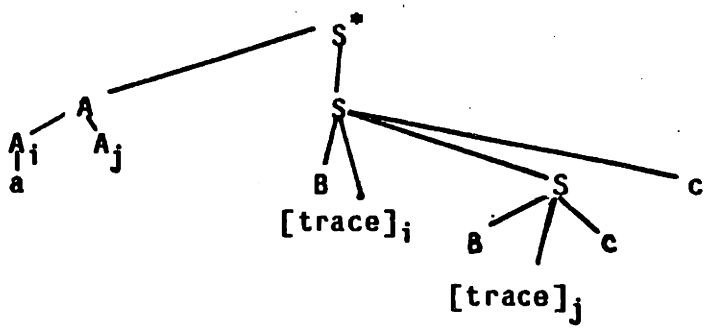
How could Parsifal handle this example? If we extend the input buffer, then there are two possibilities. First, one could simply use the *switch* operator directly, interchaning A's and B's in the input stack. This analysis would mimic the parser just described above. Second, one could use traces to link up constituents in the proper manner. Let us consider in some detail how this might work. First we have to come up with a reasonable annotated surface structure for the language. One possibility is to generate a base structure like this:



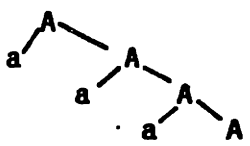
And then displace the *a*'s to the front, like a kind of *wh*-fronting, by "raising" *a*'s successively and adjoining them to each higher A:



In fact, just this analysis has been proposed by Wexler [1981]. This analysis could be simulated by Parsifal using the mechanism of traces, but only if certain conditions are relaxed. First, one could allow movement to take place across any number of S nodes, and generate the structure directly. This is empirically inadequate, because of the Subjacency constraint. Second, one could assume that traces can be arbitrarily "layered," and that the binding operator can examine arbitrary adjoined structure in order to find the right NP antecedent. In the case above, we would have the following structure:



But unless the adjoined structure can be searched in reverse order, there seems to be no easy way to bind the traces this way rather than, say, the other way around. That is, one would have to code the adjoined tree of A's as,



etc.

and then specify in a grammar rule that the first trace is to bind the first "a" that is unmarked, in the process marking it. Then, when the second trace must be bound, the binding operator will again look for the first unmarked "a," this time the second one, bind it, and then label it as utilized. Note that this is an expansion of the power of the binding operator to use an auxiliary stack, in this case, the tree of A's. It is therefore not surprising that this addition can simulate the effect of an arbitrarily large input buffer stack, as in the straight-forward approach described above.

Wexler [1981] shows that the language $a^n b^n b^n$ is learnable, given the assumptions of the Wexler-Culicover "Degree-2" theory. Therefore, the constraints imposed by the Degree-2 theory are weaker than those imposed by Marcus. In fact, as mentioned in Chapter 2, it is obvious just why this should be. It is easy to acquire grammars that are ambiguous, and these cannot have deterministic parsing procedures. Conversely, if a grammar meets the local parsability criteria necessary for LR(k,t) parsing, then that grammar satisfies the Degree-2 Bounded Degree of Error condition. Therefore, it is learnable in the Degree-2 theory. This result is not paradoxical. It simply means that one can acquire a system of knowledge whose consequences are not necessarily computationally tractable, just as one can know arithmetic, but not all the theorems of arithmetic.

5.3 Bounded Degree of Error

To conclude this chapter, we will discuss the connections between Lparsifal other current formalizations of transformational grammar, in particular the Wexler-Culicover Degree-2 theory and the Lasnik and Kupin [1977] theory.

Wexler and his colleagues have developed a specific mathematical model of the acquisition of transformational grammar. In so doing, they found that in order to guarantee learnability they had to impose a number of principles on transformational rule functioning. These are summarized in the table below, reprinted from Chapter 1. Basically, two kinds of constraints were advanced: (1) constraints on the application of a single rule; and (2) constraints on the interaction of rules from one S domain to the next.

The constraint on single rules is easy to understand: no single rule can refer to unbounded context. It is clear just why this constraint is necessary. If it were not imposed, then a single rule could refer to an arbitrarily large context. Not only would this be a difficult situation for parsing, but since an arbitrarily context would have only a small chance of appearing, approaching zero, the rule context would not be encountered, and the rule would not be acquired. This bound is enforced by essentially the Subjacency constraint, though Wexler and Culicover discovered this principle independently of Chomsky. The effect of Subjacency is to limit rule context to just the current S that is being analyzed, plus the next higher S. This constraint has a natural and direct interpretation in the Marcus parser. Grammar rules can refer only to their current active node (possibly an S) and the current cyclic node above that (also possibly an S).[11] Therefore grammar rules cannot refer to unbounded context -- in fact, they must trigger on precisely the same domain as the operations spelled out by Subjacency.[12]

The constraints on rule interactions are more complex. Basically, one must guarantee that there are a finite number of possible inter-cyclic interactions. There are two possible classes of such interactions: (1) an interaction of material: a transformation can move an entire lower context into a higher context; and (2) an interaction of context: a transformation can be triggered or inhibited by the context of another cycle. Direct (single rule) effects have already been considered. How could a series of rules cause interactions? A multiple rule chain could be arbitrarily long, and hence provide unbounded context to trigger or inhibit a rule arbitrarily far away. This clearly would have the same undesirable effect as a single unbounded context. Thus our goal will be to limit the amount of material that can be passed from one S cycle to the next. There are several constraints that are required here.

---

11. Recall that here we have modified the Marcus definition so that Subjacency is enforced directly by the parser, in effect.
12. Again, this leaves a problem with unbounded PRO indexing.

First, since a transformation can move an entire tree, an arbitrary context could be moved in a series of jumps, only to be torn apart later on. To rule out the movement of arbitrary context, the Degree-2 theory advances constraints that in effect render a phrase inaccessible to further syntactic action after it has been moved across an S boundary. Again, there are two senses of "accessible" here: the tree itself might be torn apart or syntactically manipulated; or it might be used as the context for some other rule. These two possibilities are ruled out by separate constraints posited by the Degree-2 theory. The Raising Principle rules out syntactic manipulation. After a tree has been moved across an S boundary, its internal syntactic constituents cannot be moved by any later rules. Now observe that this is precisely what goes on in the Marcus parser after a node is *lowered* into another S. This is because, by assumption, a node can be lowered into another clause only by first dropping that node into the input buffer, then creating the new clause node (an S), and finally attaching the node to be lowered to the new clause node. But this means that the node that was lowered had to have been completed, as in always the case, so that it could be dropped into the input buffer. By the assumption of opacity discussed at the end of Chapter 3, this means that all the internal details of the node so lowered cannot be altered; that material has already been attached to the NP. Thus the Marcus parser enforces the Raising Principle.[13]

The constraint on context is dubbed the No Bottom Context constraint. It says, roughly, that if a transformation acts across an S domain, moving an element from a lower S to a higher S, then it cannot refer to context in the lower S in order to act. This requirement is imposed so that material raised from yet another prevous (lower) S cannot be used as a triggering context, thereby creating a situation where context material can be successively "cascaded" from one domain to the next. Does Parsifal obey this constraint? Suppose it did not. Then the analogue of "raising" a node, namely, lowering a node, could be triggered by some arbitrary context to the right of that node. But this would amount to a violation of the the LR(k,t) condition. Therefore, Parsifal does obey this constraint.

There are other restrictions advanced in the Degree-2 theory that play a role in establishing the learnability of a transformational system, but these will not be discussed here.

---

13. Williams [1981d] shows that the Raising Principle can be strengthened to do nearly all the work of a second principle advanced by Wexler and Culicover to eliminate unbounded syntactic manipulation, namely, the Freezing Principle.

Wexler and Culicover Constraints	Acquisition Procedure Constraints
Incremental rule acquisition	Incremental rule acquisition
Universal base (can be weakened assuming a theory of base rule base rule acquisition)	Universal base (can be weakened assuming a theory of base rule acquisition)
NO negative external evidence	NO negative external evidence
Only current sentence used to construct new rule	Only current sentence used to construct new rule
Small number of new rules available for hypothesis	Small number of new rules available for hypothesis
Rule construction based on "simple" data: depth of embedding at most two	Rule construction based on "simple" data: depth of embedding at most two
Binary principle	
Freezing principle	Determinism plus locality restrictions imposed by buffer and active node stack
Raising principle	

Figure 5.3 - Constraints advanced by Wexler and Culicover vs. those of the acquisition procedure.

Finally, let us compare the constraints of Lparsifal with those independently advanced in Lasnik and Kupin's formalization of transformational grammars. We simply observe here that the restrictions are quite close.

Lasnik and Kupin Constraints	Lparsifal Constraints
Only one action per rule, affecting at most two constituents	Only one action per rule, affecting at most two constituents
Rules not marked as option or obligatory (no extrinsic ordering	Rules not marked as optional or obligatory (no extrinsic ordering
Rule patterns use only one string condition; no arbitrary Boolean conditions	Rule patterns use only one string condition; no arbitrary Boolean conditions
Subjacency	Subjacency
More specific rules trigger before more general rules	More specific rules trigger before more general rules
Small number of actions	Small number of actions

Figure 5.4 - Comparison of Lasnik and Kupin constraints to those of Lparsifal.

References

Aho, A. and Ullman, J. [1972] *The Theory of Translation, Parsing, and Compiling*, Englewood-Cliffs, NJ: Prentice-Hall.

Aho, A. and Ullman, J. [1977] *Principles of Compiler Design*, Reading, MA: Addison-Wesley.

Akmajian, A. Steele, S. and Wasow, T. [1979] "The Category AUX in Universal Grammar," *Linguistic Inquiry* 10:1.

Alperin, L. [1982] *A Parser for Japanese Sentences*, B.S. Dissertation, MIT Department of Electrical Engineering and Computer Science.

Anderson, J. [1977] "Induction of Augmented Transition Networks," *Cognitive Science*, 1.

Angluin, D. [1978] "Inductive Inference of Formal Languages from Positive Data," *Information and Control*, 45.

Aoun, J. [1979] "Indexing and Constituency," mimeographed, MIT Department of Linguistics.

Baker, C.L. [1979] "Syntactic Theory and the Projection Problem," *Linguistic Inquiry*, 10:4.

Baker [1981], "Learnability and the English Auxiliary System," in C.L. Baker and J.J. McCarthy (eds.) *The Logical Problem of Language Acquisition*, Cambridge, MA: MIT Press.

Baltin, M. [1978] *Toward a Theory of Movement Rules*, PhD Dissertation, MIT Department of Linguistics and Philosophy.

Berwick, R. [1980] "Computational Analogues of Constraints on Grammars: A Model of Syntactic Acquisition," *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA.

Berwick, R. [1981] "Computational Complexity and Lexical-Functional Grammar," *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*, Stanford, CA.

Berwick, R. and Weinberg, A. [1981a] "Parsing Theory, Computational Complexity, and the Evaluation of Grammatical Theories," *Linguistic Inquiry*, 13:2.

Berwick, R. and Weinberg, A. [1981b] "The Role of Grammars in Models of Language Use," to appear, *Cognition*.

Berwick, R. and Weinberg, A. [1982 forthcoming] *Linguistic and Computational Theory*, Cambridge, MA: MIT Press.

Berwick, R. and Wexler, K. [1982] "Parsing Efficiency, Binding, and C-command," *Proceedings of the First Annual West Coast Conference on Formal Linguistics*, Stanford, CA.

Biermann, A. and Feldman, J. [1972], "On the Synthesis of Finite State Machines from Samples of Their Behavior," *IEEE Transactions on Computation*, Vol. C-21, June, 1972.

Biermann, A. [1972] "On the Inference of Turing Machines from Sample Computations," *Artificial Intelligence*, 3.

Blum, M. [1967] "A Machine-independent Theory of the Complexity of Recursive Functions," *Journal of the Association for Computing Machinery*, 14.

Book, R. [1970] "Time-bounded grammars and Their Languages," *Journal of Computer and System Science*, 5.

Braine, M. [1971] "On Two Types of Models of the Internalization of Grammars," in D. Slobin (ed.), *The Ontogenesis of Grammar: A Theoretical Symposium*, New York: Academic Press.

Brayer and Fu [1977] "A Note on the K-tail Method of Tree Grammar Inference," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7.

Bresnan, J. [1972] *Theory of Complementation in English Syntax*, PhD Dissertation, MIT Department of Linguistics.

Bresnan, J. [1976] "On the Form and Functioning of Transformations," *Linguistic Inquiry*, 7:1.

Bresnan, J. [1978] "A Realistic Transformational Grammar," in M. Halle, J. Bresnan, and G. Miller, *Linguistic Theory and Psychological Reality*, Cambridge, MA: MIT Press.

Bresnan, J. [1982 forthcoming] *The Mental Representation of Grammatical Relations*, Cambridge, MA: MIT Press.

Bresnan, J. and Kaplan, R. [1980] "A Formal System for Grammatical Representation," in J. Bresnan, (ed.) *The Mental Representation of Grammatical Relations*, Cambridge, MA: MIT Press.

Brown, R. and Hanlon C. [1970] "Derivational Complexity and Order of Acquisition in Child Speech," in J.R. Hayes (ed.) *Cognition and the Development of Language*, New York: John Wiley.

Brown, R. [1973] *A First Language: the Early Stages*, Cambridge, MA: Harvard University Press.

Buneman, P. and Levy, L. [1978] "Proper Analysis Algorithms," in *Proceedings of the 1978 Conference on Information Sciences and Systems*, Johns Hopkins University, Baltimore, Md.

Chaitin, G. [1975] "A Theory of Program Size Formally Identical to Information Theory," *Journal of the Association for Computing Machinery*, 22.

Chomsky, N. [1951] *Morphophonemics of Modern Hebrew*, M.S. Dissertation, University of Pennsylvania Department of Linguistics. (Reprinted by Garland Press).

Chomsky, N. [1955, printed 1975] *The Logical Structure of Linguistic Theory*, New York: Plenum Press.

Chomsky, N. [1956] "Three Models for the Description of Language," *Proceedings of the Group on Information Theory*, 2:3.

Chomsky, N. [1959] "On Certain Formal Properties of Grammars," *Information and Control*, 2; in R.D. Luce, R. Bush, and E. Galanter (eds.) *Readings in Mathematical Psychology*, Vol. 2, New York: John Wiley, 1963.

Chomsky, N. [1965] *Aspects of the Theory of Syntax*, Cambridge, MA: MIT Press.

Chomsky, N. [1970] "Remarks on Nominalization." in R.A. Jacobs and P.S. Rosenbaum, *Readings in Transformational Grammar*, Waltham, MA: Blaisdell.

Chomsky, N. [1975]*Reflections on Language*, New York: Pantheon.

Chomsky, N. [1980], "On Binding," *Linguistic Inquiry* 11:1.

Chomsky, N. [1981] *Lectures on Government and Binding*, Dordrecht: Foris Publications.

Chomsky, N. and Halle, M. [1968] *The Sound Pattern of English*, New York: Harper and Row.

Chomsky, N. and Miller, G. [1957] *Pattern Conception*, Report AFRC-TN-57.

Chomsky, N. and Miller, G. [1963] "Finitary Models of Language Users," in R.D. Luce, R. Bush, and E. Galanter (eds.) *Readings in Mathematical Psychology*, Vol. 2, New York: John Wiley.

Cobham, A. [1964] "The Intrinsic Computational Difficulty of Functions" in *Proceedings of the 1964 Congress for Logic, Mathematics, and the Philosophy of Science*, Amsterdam, Holland.

Cook, S. [1981] "Towards a Complexity Theory of Synchronous Parallel Computation," University of Toronto, Department of Computer Science.

Crespi-Reghizzi, S. [1970] *The Mechanical Acquisition of Precedence Grammars* UCLA ENG-7054.

Crespi-Reghizzi, S. [1971] "An Effective Model for Grammar Inference," *Proceedings of the IFIP Congress 71*, Amsterdam: North-Holland.

Crespi-Reghizzi, S. Guida, G. Mandrioli, D. [1978] "Non-counting Context-free Languages," *Journal of the Association for Computing Machinery* 25:4.

Culicover, P. and Rochemont, M. [1980] *Stress and Focus in English*, SCSI, University of California at Irvine Studies in the Cognitive Sciences, Irvine, CA.

De Remer, F. [1969] *Practical Translators for LR(k) Languages* PhD Dissertation, MIT Department of Electrical Engineering and Computer Science, Laboratory for Computer Science Report MAC-TR-65.

Earley, J. [1968] *An Efficient Context-free Parsing Algorithm*, PhD Dissertation, Carnegie-Mellon University, Department of Computer Science.

Emonds, J. [1969] "A Structure-Preserving Constraint on NP Movement Transformations," in R. Binnick, A. Davison, G. Green, and J. Morgan, (eds.) *Papers from the Fifth Regional Meeting of the Chicago Linguistic Society*, Chicago: University of Chicago.

Evey, R. [1963] "Theory and Application of Pushdown-store Machine," *AFIPS Fall Joint Computer Conference*, 24. New York: Spartan.

Farmer, A. [1980] *On the Interaction of Morphology and Syntax* PhD Dissertation, MIT Department of Linguistics.

Feldman, J. [1972] "Some Decidability Results on Grammatical Inference and Complexity," *Information and Control*, 20.

Feldman, J. Gips, J. Horning, J., Reder, S. [1969] *Grammatical Complexity and Inference* Stanford AI-89, CS125.

Feldman, J. and Shields, P. [1970] "Total Complexity and the Inference of Best Programs," *Mathematical Systems Theory*, 10.

Fiengo, R. [1974] *Semantic Conditions on Surface Structure*, PhD Dissertation, MIT Department of

Linguistics.

Fiengo, R. [1977] "On Trace Theory," *Linguistic Inquiry*, *:1.

Fillmore, C. [1968] "The Case for Case," in E. Bach and R. Harms (eds.) *Universals in Linguistic Theory*, New York: Holt, Rinehart, and Winston.

Fischer, M. [1969] "Some Properties of Precedence Languages," *Proceedings of the ACM Symposium on the Theory of Computation.*

Fletcher, P. [1979] "The Development of the Verb Phrase," in P. Fletcher and M. Garman (eds.), *Language Acquisition*, New York: Cambridge University Press.

Floyd, R. [1963] "Syntactic Analysis and Operator Precedence," *Journal of the Association for Computing Machinery*, 10:3.

Fodor, J.D. [1978] "Parsing Strategies and Constraints on Transformations," *Linguistic Inquiry*, 9:3

Fodor, J.D. [1980] "Superstrategy," in W. Cooper and E. Walker, *Sentence Processing: Psycholinguistic Studies Presented to Merrill Garrett*, Hillsdale, NJ: Erlbaum.

Fodor,J.A., Bever, T., and Garrett, M. [1974] *The Psychology of Language* New York: McGraw-Hill.

Ford, M., Bresnan, J., and Kaplan, R. [1981] *Toward a Theory of Lexico-Syntactic Interactions in Sentence Processing* in J. Bresnan (ed.) *The Mental Representation of Grammatical Relations*, Cambridge, MA: MIT Press.

Frazier, L. [1979] *On Comprehending Sentences: Syntactic Parsing Strategies*, PhD Dissertation, University of Massachusetts Department of Linguistics, Amherst.

Fu, K. and Booth, T. [1975] "Grammatical Inference: Introduction and Survey," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-5(1), SMC-5(4).

Gazdar, G. [1980] *Phrase Structure Grammar*, in P. Jacobson and G. Pullum (eds.) *The Nature of Syntactic Representation.*

Gazdar, G. [1981] "Unbounded Dependencies and Co-ordinate Structure," *Linguistic Inquiry* 12:2.

Ginsburg, S. and Lynch, N. [1976] "Size Complexity in Context-free Grammar Forms," *Journal of the Association for Computing Machinery*, 23.

Ginsburg, S. and Rose, G. [1966] "Preservation of Languages by Transducers," *Information and Control*, 9.

Gold, E. [1967] "Language Identification in the Limit," *Information and Control*, 10.

Gold, E. [1978] "Complexity of Automaton Identification from Given Data," *Information and Control* 37.

Gonzalez, R. and Thomason, M. [1978] *Syntactic Pattern Recognition*, Reading, MA: Addison-Wesley.

Gray, J. [1969] *Precedence Parsers for Programming Languages*, PhD Dissertation, University of California at Berkeley Department of Computer Science.

Greenberg, J. [1963] *Universals of Language*, Cambridge, MA: MIT Press.

Grimshaw, J. [1981] "Form, Function, and the Language Acquisition Device," in C.L. Baker and J.J. McCarthy (eds.) *The Logical Problem of Language Acquisition*, Cambridge, MA: MIT Press.

Hale, K. [1979] "On the Position of Warlpiri in a Typology of the Base," mimeographed, MIT.

Halle, M. [1961] "On the Role of Simplicity in Linguistic Descriptions," *Proceedings of Symposia in Applied Mathematics*, 12.

Hamburger, H. and Wexler, K. [1975] "A Mathematical Theory of Learning Transformational Grammar," *Journal of Mathematical Psychology*, 12.

Hammer, M. [1974] *A New Transformation into Deterministic Top-Down Form*, PhD Dissertation, Department of Electrical Engineering and Computer Science.

Harman, G. [1963] "Generative Grammars without Transformation Rules: in Defense of Phrase Structure," *Language*, 39.

Harrison, M. [1978] *Introduction to Formal Language Theory*, Reading, MA: Addison-Wesley.

Hartmanis, J. and Stearns, R. [1965] "On the Computational Complexity of Algorithms," *Transactions of the American Mathematical Society*, 117.

Heny F. [1979] "Review of *The Logical Structure of Linguistic Theory*," *Synthese*, 40.

Hopcroft, J. and Ullman, J. [1979] *Introduction to Automata Theory, Languages, and Computation*,,

Reading, MA: Addison-Wesley.

Horning, J. [1969] *A Study of Grammatical Inference*, Stanford AI Report CS 139.

Horning, J. [1971] "A Procedure for Grammatical Inference," *Proceedings of the IFIP Congress 71*, Amsterdam: North-Holland.

Hornstein, N. [1977] "S and the X Convention," *Linguistic Analysis*, 3:2.

Iba, G. [1979] *Learning Disjunctive Concepts from Examples*, MIT Artificial Intelligence Laboratory Memo 548.

Jackendoff, R. [1977] *X-Syntax: A Study of Phrase Structure*, Cambridge, MA: MIT Press.

Jakobson, R. and Halle, M. [1956] *Fundamentals of Language*, The Hague: Mouton.

Jakobson, R. [1962] *Selected Writings I*, The Hague: Mouton.

Jakobson, R. [1968] *Child Language, Aphasia, and Phonological Universals*, The Hague: Mouton.

Joshi, A. and Levy, L. [1977a] "Skeletal Structural Descriptions," *Information and Control*, 39.

Joshi, A. and Levy, L. [1977b] "Constraints on Structural Descriptions: Local Transformations," *SIAM Journal of Computing*, 6.

Joshi, A., Levy L., and Yueh, K. [1980] "Local Constraints on Programming Languages," *Journal of Theoretical Computer Science*, 12:3.

Kaplan R. [1974] *Transient Processing Load in Relative Clauses*, PhD Dissertation, Harvard University Department of Psychology and Social Relations.

Katz, J. [1966] *The Philosophy of Language*, New York: Harper and Row.

Kean, M. [1974] *The Theory of Markedness in Generative Grammar*, PhD Dissertation, MIT Department of Linguistics.

Keil, F. [1979] *Semantic and Conceptual Development: An Ontological Perspective*, Cambridge, MA: Harvard University Press.

Kiparsky, P. [1973] "Elsewhere in Phonology," in S. Anderson and P. Kiparsky (eds.), *A Festschrift for Morris Halle*, New York: Holt, Rinehart, and Winston.

Knuth, D. [1965] "On the Translation of Languages from Left to Right," *Information and Control*, 8.

Kolmogorov, A. [1965] "Three Approaches to the Quantitative Definition of Information," *Problems of Information Transmission*, 1.

Koster, J. [1978] *Locality Principles in Syntax*, Dordrecht: Foris Publications.

Kuno, S. *The Structure of the Japanese Language*, Cambridge, MA: MIT Press.

Labov, W. [1970] "The Study of Language in its Social Context," *Studium Generale*, 23.

Labov, W. and Labov, T. [1976] "Learning the Syntax of Questions," *Proceedings of the Conference on the Psychology of Language, Scotland*, July 1976.

Langendoen, T. [1975] "Finite-State Parsing and the Status of Readjustment Rules," *Linguistic Inquiry*,

Langley, P. [1977] *BACON: A Production System that Discovers Empirical Laws*, Carnegie-Mellon University, CIP WP 360.

LaPointe, S. [1980] "A Lexical Analysis of the English Auxiliary Verb System," in T. Hoekstra, H. van der Hulst, and M. Moortgat (eds.), *Lexical Grammar*, Dordrecht: Foris Publications.

Lasnik, H. and Kupin, J. [1977] "A Restrictive Theory of Transformational Grammar," *Theoretical Linguistics*, 4:3.

Lasnik, H. [1980] "Restricting the Theory of Transformations," in D. Lightfoot and N. Hornstein (eds.), *Explanation in Linguistics*.

Lasnik, H. [1981] "Learnability, Restrictiveness, and the Evaluation Metric," in C. Baker and J. McCarthy (eds.) *The Logical Problem of Language Acquisition*, Cambridge, MA: MIT Press.

Leopold, W. [1949] *Speech Development of a Bilingual Child: A Linguists' Record* vol.3, Evanston, IL: Northwestern University Press.

Levine, B. [1981] "Derivatives of Tree Sets with Applications to Grammatical Inference," *IEEE Transactions on Pattern Matching and Machine Intelligence*, 3:3.

Liberman, M. and Prince, A. [1977] "On Stress and Linguistic Rhythm," *Linguistic Inquiry*, 8:2

Lieber, S. [1980] *On the Organization of the Lexicon*, PhD Dissertation, MIT Department of

Linguistics and Philosophy.

Lightfoot, D. and Hornstein, N. [1981] "Introduction," in D. Lightfoot and N. Hornstein (eds.), *Explanation in Linguistics*, New York: Longmans.

Limber, J. [1973] "The Genesis of Complex Sentences," in T. Moore (ed.) *Cognitive Development and the Acquisition of Language*, New York: Academic Press.

Machtey, M. and Young, P. [1978] *An Introduction to the General Theory of Algorithms*, New York: North-Holland.

Maling. J. and Zaenen, A. [1980] "A Base-generated Account of Extraction Phenomena in Scandanavian Languages," in P. Jacobson and G. Pullum (eds.) *The Nature of Syntactic Representation.*

Marantz, A. [1981] *On the Nature of Grammatical Relations*, PhD Dissertation, MIT Department of Linguistics and Philosophy.

Marantz, A. [1980] "Assessing the X Convention," mimeo, MIT.

Maratsos, M. [1978] "New Models in Linguistics and Language Acquisition," in J. Bresnan, M. Halle, and G. Miller, *Linguistic Theory and Psychological Reality*, Cambridge, MA: MIT Press.

Marcus, M. [1977] *A Theory of Syntactic Recognition for Natural Language*, PhD Dissertation, MIT Department of Electrical Engineering and Computer Science.

Marcus, M. [1980] *A Theory of Syntactic Recognition for Natural Language*, Cambridge, MA: MIT Press.

Marr, D. and Nishihara, K. [1978] "Visual Information Processing and the Sensorium of Sight," *Technology Review*, October.

Marr, D. and Poggio, T. [1977] "From Understanding Computation to Understanding Neural Circuitry," *Neuroscience Research Program Bulletin*, 15:3.

Marshall, J. [1979] "Language Acquisition in a Biological Framework," in P. Fletcher and M. Garman (eds.), *Language Acquisition*, New York: Cambridge University Press.

Mayer, J., Erreich, A. and Valian, V. [1978] "Transformations, Basic Operations, and Language Acquisition," *Cognition*, 6.

McCreight, E. and Meyer, A.R. [1969] "Classes of Computable Functions Defined by Bounds on Computation," *Proceedings of the ACM Symposium on the Theory of Computing.*

McDermott, J. and Forgy, C. [1978] "OPS: A Domain Independent Production System Language," in D. Waterman and F. Hayes-Roth, *Pattern-Directed Inference Systems*, New York: Academic Press.

McNeil, D. [1966] "Developmental Psycholinguistics," in F. Smith and G. Miller (eds.) *The Genesis of Language*, Cambridge, MA: MIT Press.

Meyer, A. and Fischer, M. [1971] "Economy of Description by Automata, Grammars, and Formal Systems," *12th Annual Symposium on Switching and Automata Theory.*

Mitchell, T. [1978] *Version Spaces: An Approach to Concept Learning*, PhD Dissertation, Stanford University Department of Computer Science.

Morgan, J. and Newport, E. [1981] "The Role of Constituent Structure in the Induction of an Artificial Language," *Journal of Verbal Learning and Verbal Behavior*, 20.

Newport, E., Gleitman, L. and Gleitman, H. [1977] "Mother, I'd Rather do it Myself: Some Effects and Non-effects of Maternal Speech Style," in C. Snow and C. Ferguson (eds.) *Talking to Children, Input and Acquisition*, New York: Cambridge University Press.

Olivier, D. [1968] *Stochastic Grammars and Language Acquisition Mechanisms*, PhD Dissertation, Harvard University Department of Psychology and Social Relations.

Olmsted, D. [1971] *Out of the Mouth of Babes: Earliest Stages in Language Learning*, The Hague: Mouton.

Peters, S. and Ritchie, R. [1969] "Context-sensitive Immediate Constituent Analysis: Context-free Languages Revisited," *Proceedings of the ACM Symposiyum on the Theory of Computation.*

Peters, S. and Ritchie, R. [1973] "On the Generative Power of Transformational Grammars," *Information Sciences*, 6.

Pinker, S. [1979] "Formal Models of Language Learning," *Cognition*, 7:3.

Pinker, S. [1981] "A Theory of the Acquisition of Lexical-Interpretive Grammars," in J. Bresnan (ed.), *The Mental Representation of Grammatical Relations* Cambridge, MA: MIT Press.

Plotkin, G. [1970] "A Note on Inductive Generalization," in B. Meltzer and D. Michie (eds.) *Machine*

Intelligence v. 5. Edinburgh: Edinburgh University Press.

Ritchie, R. [1963] "Classes of Predictably Computable Functions," *Transactions of the American Mathematical Society*, 106.

Rizzi, L. [1978] "A Restructuring Rule in Italian Syntax," in S. Keyser (ed.), *Recent Transformational Studies in European Linguistics*, Cambridge, MA: MIT Press.

Rizzi, L. [1980] "Comments on Chomsky," in J. Mehler (ed.) *Proceedings of the June 1980 Conference on the Cognitive Sciences*, Paris.

Roeper, T. [1980] *Handout at Sloan Conference, MIT, 1980.*

Roeper, T. [1981] "On the Deductive Model and the Acquisition of Productive Morphology," in C. Baker and J. McCarthy (eds.) *The Logical Problem of Language Acquisition*, Cambridge, MA: MIT Press.

Rosenberg, A. [1967] "Real-time Definable Languages," *Journal of the Association for Computing Machinery*, 14:4.

Ross, J. [1967] *Constraints on Variables in Syntax*, PhD Dissertation, MIT Department of Linguistics.

Rounds, W. [1975] "A Grammatical Characterization of the Exponential-Time Languages," *Proceedings of the 16th Symposium on Switching and Automata Theory.*

Rychener, M. and Newell, A. [1977] "An Instructible Production System: Basic Design Issues," in *Pattern-Directed Inference Systems*, New York: Academic Press.

Shipman, D. [1979] *Phrase Structure Rules for Parsifal*, MIT Artificial Intelligence Laboratory Working Paper 182.

Small, S. *Word Expert Parsing*, PhD Dissertation, University of Maryland.

Sober, E. [1975] *Simplicity*, London: Oxford University Press.

Solomonoff, R. [1964] "A Formal Theory of Inductive Inference," *Information and Control*, 7:1.

Sommers, F. [1971] "Structural Ontology," *Philosphia* 1.

Stowell, T. [1981] "Origins of Phrase Structure," PhD Dissertation, MIT Department of Linguistics and Philosophy.

Swartout, W. [1978] *A Comparison of Parsifal with Augmented Transition Networks*, MIT Artificial Intelligence Laboratory, Memo 462.

Szymanski, T. and Williams, J. [1976] "Non-canonical Extensions of Bottom-up Parsing Techniques," *SIAM Journal of Computing* 5.

Tavakolian, S. [1977] *Structural Principles in the Acquisition of Complex Sentences* PhD Dissertation, Department of Linguistics, University of Massachusetts, Amherst.

Thatcher [1973] "Tree Automata: An Informal Survey," in A. Aho (ed.) *Currents in the Theory of Computing*, Englewood Cliffs, NJ: Prentice-Hall.

Thiersch, C. [1978] *Topics in German Syntax*, PhD Dissertation, MIT Department of Linguistics and Philosophy.

Turnbull, C. [1975] *Deterministic Left to Right Parsing*, CSRG-46, Toronto: University of Toronto Computer Systems Research Group.

Vergnaud, J. and Halle, M. [1980] "Three Dimensional Phonology," *Journal of Linguistic Research*, 1:1.

Walters, D. [1970] "Deterministic Context-sensitive Languages," *Information and Control*, 17:1.

Wanner, E. and Gleitman, L. [1982 forthcoming] *Language Acquisition -- the State of the Art.*

Weinberg, A. and Hornstein, N. [1981] "Case Theory and Preposition Stranding," *Linguistic Inquiry*, 12:1.

Weinberg, A. [1982] *Passives, Datives, and the Theory of Language Acquisition*, unpublished ms, MIT.

Wexler, K. and Culicover, P. [1980] *Formal Principles of Language Acquisition*, Cambridge, MA: MIT Press.

Wharton, R. [1977] "Grammar Enumeration and Inference," *Information and Control*, 33.

Williams, E. [1977] "Discourse and Logical Form," *Linguistic Inquiry*, 8:1.

Williams, E. [1981a] "Predication," *Linguistic Inquiry*, 11:1.

Williams, E. [1981b] "On the Notions 'Lexically Related' and 'Head of Word'", *Linguistic Inquiry*, 12:2.

Williams, E. [1981c] "Language Acquisition, Markedness, and Phrase Structure," in S. Tavakolian (ed.), *Language Acquisition and Linguistic Theory*, Cambridge, MA: MIT Press.

Williams, E. [1981d] "A Readjustment in the Learnability Assumptions," in C. Baker and J. McCarthy (eds.) *The Logical Problem of Language Acquisition*, Cambridge, MA: MIT Press.

Williams, J. [1975] "Bounded-context Parsable Grammars," *Information and Control*, 28.

Winston, P. [1975] "Learning Structural Descriptions from Examples," in P. Winston (ed.) *The Psychology of Computer Vision*, Reading, MA: Addison-Wesley.

Wirth, N. and Weber, H. [1966] "Euler -- a Generalization of ALGOL and Its Formal Definition," *Communications of the Association for Computing Machinery*, 9:1 and 9:2.

Woods, W. [1970] "Transition Network Grammars for Natural Language Analysis," *Communications of the Association for Computing Machinery*, 13:10.

Yngve, V. [1960] "A Model and an Hypothesis for Language Structure," *Proceedings of the American Philosophical Society*, 104.