UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AI-TR 972	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Principle-Based Parsing		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert C. Berwick		8. CONTRACT OR GRANT NUMBER(s) NSF Grant#DCR-8552543, NSF Grant DCR-8511531-IST ARPA/ONR Contract#N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE June, 1987
		13. NUMBER OF PAGES 113 (including coverpage)
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Natural Language Processing
Language Translation
Rule-Based Systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

During the past few years, there has been much discussion of a shift from rule-based systems to principle-based systems for natural language processing. This paper outlines the major computational advantages of principle-based parsing, its differences from the usual rule-based approach, and surveys several existing principle-based parsing systems used for handling languages as diverse as Warlpiri, English, and Spanish, as well as language translation.

Technical Report 972

Principle-Based Parsing

Robert C. Berwick

MIT Artificial Intelligence Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. T.R. No. 972 June, 1987

PRINCIPLE-BASED PARSING

Robert C. Berwick

ABSTRACT

During the past few years, there has been much discussion of a shift from *rule-based* systems to *principle-based* systems for natural language processing. This paper outlines the major computational advantages of principle-based parsing, its differences from the usual rule-based approach, and surveys several existing principle-based parsing systems used for handling languages as diverse as Warlpiri, English, and Spanish, as well as language translation.

Acknowledgments: This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research has been provided in part by NSF Grant DCR-8552543, under a Presidential Young Investigator Award to Professor Robert C. Berwick, in part by NSF Grant DCR-8511531-IST, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

(To appear in *The Processing of Linguistic Structure*, MIT Press, 1987)

1 Introduction

In the past few years, there has been much talk of a dramatic shift away from rules in linguistic theories to systems of interacting principles. Government-binding (GB) theory has, perhaps, been regarded as leading this charge: for example, as should be familiar to many by now, there are no particular transformational "rules" in GB theory. On this recent view, there is no single "rule" of passive; rather, passive constructions result from the interactions of deeper morphological and syntactic operations (case absorption, thematic role assignment, and free movement).[1] These interacting principles happen to have, as surface realizations, sentence forms that may be described as "active" and "passive," and were so described in earlier theories, for example, by transformations, or even by pairs of context-free rules related by metarules.

But this shift is certainly not unique to GB theory. Many other current linguistic theories, among them generalized phrase structure grammar (and head-driven phrase structure grammar), have gradually led in the direction of systems of declarative constraints that are not construction specific. For instance, modern GPSG theory full of principles that fix the well-formedness of surface strings without spelling out, explicitly, their detailed phrase structure: the Foot Feature Principle, the Head Feature Convention, Control Agreement Principle, and so forth replace explicit systems of phrase structure rules with constraint sets. These theories too no longer contain large numbers of particular rules, but schemas constrained according to syntactic and morphological principles.

We may contrast a principle-based approach with a *construction-based* approach to language: the difference is that a principle-based approach aims to reconstitute the vocabulary of grammatical theory in such a way that surface and language-particular constructions like passive follow from the interactions of a different set of primitive elements than before. In contrast, a construction-based approach attempts to describe surface sentence forms roughly as they seem to occur on the surface. Roughly and intuitively, the analogy is that doing construction-based linguistics is like doing chemistry without molecular and atomic theory: while one can describe the various chemical compounds that appear, one cannot say why certain chemicals and not others are found—this requires a new scientific vocabulary that consisting of basic elements and their principles of combination, exactly what the principle-based approach seeks.

But what exactly does this shift to principles imply for *parsing*? A whole host of important issues arise, most of which are only just now being explored. While the rest of the chapter will address these in more detail, I would like to set the stage with some more general remarks. Section 2 probes these points in more depth.

[1]See Jaeggli 1986 for an up-to-date discussion of passive constructions described this way.

First: *there is no such thing as a "GB parser"*.[2] GB is an a-computational theory of the language faculty. Therefore, it's basically a category error to say that one has a GB parser. (Cast in familiar terms: GB theory is a competence theory, not a performance theory.)

Second: However, one *can* have a GB-*based* parser or a principle-*based* parser. After all, that's simply another way of saying that the language processor parses "in the manner of the grammar G," to use Chomsky and Miller's two-decade old phrase. Naturally, everything hinges on just what we take "based" or "in the manner of" to mean.

Third: Contrary to what is often assumed, a "direct" or "principle-based" implementation of a linguistic theory need not assume a close correspondence between the formal operations of a generative grammar and the computational actions that take place in linguistic processing, . The usual assumption is that the grammatical theory ought to reproduce the clustering of properties observed in psycholinguistic experiments (Frazier 1986). But there is a subtle fallacy here: this position fails to distinguish between the *temporal* sequence of operations in parsing and the *logical sequence* of operations in a derivation. If these do not correspond, it's often said that a parser is an "indirect" implementation of a grammatical theory (and as a corollary it's claimed that the grammatical theory is correspondingly somehow lacking). On just these grounds, Frazier (1986) argues from certain experimental evidence that GB-theory does not compile enough information to be psychologically realistic. However, we'll see (section 2) that one can build a language processing system that intuitively *does* directly incorporate GB theory, even though it need not directly reflect the clusters of psycholinguistic properties we see "on the outside." In other words, psycholinguistic results may not clearly point out whether or not people use a principle-based parser.

Besides these general points, many other specific questions spring immediately to mind:

- Are parsers based on principles really any different from those based on rules? If so how?
- Are principled-based parsers more efficient that rule-based parsers? Or are they somehow "compiled" (multiplied out) into a more efficient, individual rule-like form? After all, at first blush it would seem that working directly with principles rather than surface constructions could lead to a slower parser, since one might have to laboriously recalculate, each time, the permitted surface arrangements of phrases: since principles lead to a more circuitous connection between what sequence of words actually appears on the surface and the principles beneath, principle-based parsing

[2]Or even a GPSG parser.

would seem to demand a *larger* number of computational steps to map between surface sentence and underlying form. What then, if anything, is gained by grounding parsers on principles instead of rules? (See section 2 for further discussion of this point.)

- What does it mean to say that grammatical principles are used *directly* or *indirectly* in parsing? Is it just the difference between using an axiom system rather than its theorems to deduce whether a sentence is well-formed, as suggested above? (Again see section 2 for discussion.)
- What is the range of algorithms possible for principle-based parsers? What representations, or more broadly, *information structures* must they, can they, or should they use?
- What can principle-based parsers do well that rule-based parsers cannot?

In this paper I would like to examine these and related questions, arguing that the shift from rules to principles *does* lead to important new perspectives on parsing, to new insights into the relationship between grammatical theories and parsers, and even on the engineering side, to new avenues for the construction of flexible, modular, language-independent natural language processing systems.

1.1 Rules and principles: the key differences

Before proceeding though, it is important to first clear away some possible misunderstandings about the differences between rules and principles. It should be evident that most rules could just as well be expressed as declarative principles. For instance, one could just use a definitional language like Rounds' first order formalism (LFL), or definite clause grammars (DCGs), to transform concatenative context-free rules into declarative constraints. For instance, in Rounds' formalism we would have:

$$\text{S} \rightarrow \text{NP VP} \quad \Rightarrow S(x) \Longleftrightarrow \exists yz.NP(y) \wedge VP(z) \wedge x = yz$$

On this view, a set of context-free rules expressed via definite clause grammars could just as well be called a set of principles, since indeed they fit our definition as a "set of statements, well-grounded on general empirical evidence, such that a range of phenomena follow from their interaction."

Similarly, any principle A can be converted into a procedural rule that simply says "apply principle A." Evidently then, there is nothing to be gained at this level of discussion.

Indeed, the rule-principle distinction here nearly smacks of the declarative-procedural controversy that once (and sometimes still) swirls about in A.I.: are facts about some domain best stated in a program (a procedural form), or stated declaratively? I think that pursuing a formal rule-principle distinction will be about as fruitful as this older dispute—which is to say, not very fruitful at all. But while in this superficial formal sense there seems to be little point in pursuing a distinction between rules and constraints, from a *practical* point of view—how the notion of "rule" and "principle" have been distinguished in linguistic practice—the difference looms large.

In practice though, "rules" have nearly always been used to describe language particular constructions like passive or raising-to-object, while principles (such as the A-over-A condition) have been taken to be language universal. For example, the first-order formalization of the rule expanding S given above is presumably language particular, hence, colloquially, a "rule."

Thus we seem to have at least two notions at work here: one is *language particular* vs. *non-language particular*, and the other is *construction specific* vs. *non-construction specific*. This yields four possible distinctions. The usual notion of a rule (such as passive), is language particular and construction specific. A principle, at least as usually considered in linguistics, is language universal and non-construction specific (though this is of course relative to how abstract we take the notion of a construction); a paradigm example is the principle "Move alpha". What of the non-construction specific, but language universal possibility? In modern theories, this role seems to be played by *parameterizing* universal principles: thus, a language may contain Move-alpha or not; or select the option that Heads of phrases are first (as in English) or final (as in Japanese). The remaining language universal and construction specific category would apply to a theory (such as that proposed by Bach 1965), positing language universal but construction specific rules such as *wh*-movement; for example, a language could either have *wh*-movment rules or not.

In short, in the remainder of this chapter we will take *principles* to be language universal statements, parameterized to handle the case of particular languages, and *principle-based parsers* to be those that in some sense use language universal statements to analyze sentences, that is, to associate linguistic descriptions with sentences.

A further distinction has to do with whether principles themselves can or ought to be stated declaratively or not. At least superficially, it appears that most modern syntactic theories can be stated purely declaratively, though it is not obvious that all linguistic levels of representation can be so stated.[3] If this is correct, then a declarative grammatical

[3]See Halle and Vergnaud 1987 chapters 1 and 4 for additional discussion on this point; evidently, empirical evidence suggests that phonological principles should not be stated in a purely declarative manner, but should reflect the possibility of ordering.

formulation of at least the D- and S-structure component of GB theory should be possible, though the difficulties in doing so are perhaps not fully appreciated. (Section 2 sketches some of the problems in broad outline.) For example, consider the important constraint of GB theory, as outlined in van Riemsdijk and Williams (1984:292, after Chomsky 1981), that an empty category must be "properly governed." (This is called the Empty Category Principle, or ECP.) Proper government is a declarative constraint that rules out sentence configurations such as (1) while admitting sentences such as (2):

(1) *John was tried [*trace* to be promoted]

(2) Who do you [think (that) Bill saw *trace*]

Van Riemsdijk and Williams define the relevant principle(s) this way:

Empty Category Principle: [*e*] (an empty element like a *trace* or a nonlexical pronominal) must be properly governed.

Proper government: X properly governs Y if and only f X governs Y and X is either X^0 (i.e., V, N, A, P); or NP_i, where $Y=NP_i$.

Government: X governs Y if and only if Y is contained in the maximal $\overline{X}$ projection of X, $X^{\mathbf{max}}$, and X^{max} is the smallest maximal projection containing Y, and X c-commands Y.

C-command: X c-commands Y if and only if the first branching node dominating X also dominates Y, and X does not itself dominate Y.

To illustrate how this very simple but basic constraint might be formulated in declarative terms, we could try defining proper government, at least for pairs of nodes:

$$\begin{array}{lll} \mathit{properly\_governs}(X,Y) & \leftarrow & \mathit{governs}(X,Y) \\ & & \wedge\ \text{lexical\_category(X)}\ . \end{array}$$

It's easy to see that this follows the English account. Putting aside the straightforward problem of defining the predicate *lexical_category*, we note that we must now define a new predicate, *govern*, again following van Riemsdijk and Williams; this predicate in turn requires four new predicates:

$$\begin{aligned} govern(X,Y) \quad \leftarrow \quad & maximal\_projection(X, Xmax) \\ & \wedge\ dominates(Xmax, Y) \\ & \wedge\ least\_maximal\_projection(Y, Xmax) \\ & \wedge\ c\_commands(X,Y). \end{aligned}$$

Again, we defer the definitions of the tree predicates *dominates*, *maximal_projection*, and *least_maximimal_projection*, defining only *c_command* in terms of *dominates* and *first_branching_node*:

$$\begin{aligned} c\_commands(X,Y) \quad \leftarrow \quad & first\_branching\_node\_from(X, BrNode) \\ & \wedge\ dominates(BrNode, Y) \\ & \wedge\ \neg dominates(X,Y) \\ & \wedge\ \neg dominates(Y,X). \end{aligned}$$

A complete set of declarative constraints for GB theory would involve similar conditions for each of its many modules, applied to all potentially well-formed tree structures. If this program could be carried out in full detail, then one would be able to apply the theorem-proving techniques of logic programming, developing a *provably* faithful principle-based parsing system. Section 2 briefly describes some of the parsing techniques that would be applicable.

However, many obstacles remain. For example, the proper government predicate only defines when one node properly governs another, but proper government of a particular empty category holds only if *there exists* some such X that properly governs the empty category. Indeed, while in general it's easy to write simple declarative statements for constraints that operate essentially under tree sisterhood, and therefore it's easy to write a declarative formulation of GB theory that "works" for bounded sentences of one S domain or two, it's a bit trickier to properly formulate declarative *chain constraints*—constraints that apply globally to an entire parse tree, like the ECP. We return to this matter in section 2.

A further, practical difference between rules and principles—the way they have been actually used in linguistic theories—lies with the differing conceptions of language that each entails. Since this difference is important to parsing, I will review this conceptual distinction here.

At heart, the rule-based view of language connects to traditional grammars with their emphasis on language-specific constructions as well as to formal language theory with its idea that we have some set of rules—a grammar—that generates a distinguished set of strings—the grammatical sentences— and no others. The rule-based generative-grammar/formal language theory tradition *starts* with a distinguished corpus of grammatical strings, to be described—"the language". This (extensional) set of strings thus becomes of central importance, since it serves as the touchstone for the rule system's success. A key issue, then, is where this special set comes from. It demands some justification; one cannot just pick out a set of strings at random.[4]

A principle-based theory starts from quite a different vantage point. On the constraint-based view, *all* sounds are assigned *some* description, depending upon which admissibility constraints they pass or which principles they abide by; there is no (artificial) division into "grammatical" and "ungrammatical" strings. Rather, some sounds pass all admissibility constraints (and so are dubbed fully grammatical), while others violate a range of constraints (and so vary in their apparent acceptability).[5] In a sense, then, the principle-based view rejects the traditional formal language theory account whereby the aim of the grammar is to generate a special set of strings and structural descriptions.

From this deep division in conceptual foundations a great many of the rule-based/principle-based distinctions flow, most with computational implications. Let me summarize these here; we will return to them in the sequel.

1.1.1 Rules and principles view language differently

- Rule-based systems are construction oriented; they tend to focus on string sets rather than grammars. This follows directly from their historical roots in traditional grammars formal language theory. In contrast, principle-based systems aim from the start to describe grammars, not languages; while they too must of course account for the variety of language constructions observed in natural languages, their emphasis is different. For instance, the context-free rule VP$\rightarrow$V NP says only that a Noun Phrase lies immediately to the right of a Verb in a Verb Phrase; this describes the construction. In contrast, a principle-based account strives to say *why* the Noun Phrase lies to the right (because in English the Verb case marks the NP under adjacency, and $\overline{\mathrm{X}}$ theory says that complements lie to the right) —that is, one is almost forced to push beyond description to say *why* phrases take the form they do.

[4]This was recognized as far back as Chomsky's *Logical Structure of Linguistic Theory*: here at least some argument was made that the primary grammatical corpus was related in some way to language acquisition. In most later work in generative grammar, attempts at justifying the initial set of grammatical strings were simply dropped.

[5]This is the point of view adopted in Chomsky's *Logical Structure of Linguistic Theory* and *Aspects of the Theory of Syntax*.

- Rule-based systems must be greatly altered to handle so-called ungrammatical input—by adding more rules, and often scoring functions of some kind. Since parsing time varies directly with grammar size, this has computational impact. In contrast, a principle-based system naturally accommodates ill-formed input; it is designed from the start to be able to parse any string.

- Rules of grammar have been generally taken to mean rules like S$\rightarrow$NP VP, or passive, raising, and the like. Further, the notion of derivation and rewrite rules makes phrase structure a central focus of rule-based systems. In contrast, for principle-based systems *admissibility conditions* or *licensing relations* are central. Phrase structure is derivative. For instance, in a GB-based theory, we might only have to recover what governs what, plus thematic assignments, rather than an explicit phrase structure tree; phrase structure is simply derivative from these more basic relations. Abney (1986) gives an example, assuming thematic relations plus three other basic licensing relations: *Subjecthood*, *Functional selection*, and *Modification*, illustrated in figure 1 (Abney's figure 2). Note how the phrase structure tree is not needed at all in order to fix the grammatical relationships for a sentence such as *John would like pictures of Mary*, but it may be easily recovered from these relationships if so desired. Thus, principle-based parsing turns ordinary context-free parsing on its head: its primary job is to recover whatever information structures are necessary build up grammatical relations, and these are not necessarily the same as tree structures. As we shall see, this difference has enormous implications for the range of parsing algorithms one can think of for principle-based systems; it opens up a world of parsing possibilities that would not be thought of with phrase-structure based systems.

- (Context-free) rule-based parsers have trouble handling so-called free-word order languages in a linguistically grounded way, while principle-based parsers can tackle them successfully. Since rules have often meant context-free rules, and since context-free rules encode both linear precedence and hierarchical information in a *single* format, such systems cannot easily tease apart the spectrum of free-word order variation found across natural languages. For instance, in Warlpiri, morpheme order is fixed, and hence precedence information needs to be recorded here; while phrasal ordering is by and large free, and so only hierarchical information needs to be used at the phrasal level. By parceling out distinct informational constraints among distinct principles and representations, a principle-based theory can separate morpheme precedence from phrasal hierarchy, and describe a language that has fixed morpheme order and free phrasal order. (We will go through this example in more detail in section 3.)

- A rule-based parser does not easily yield an account of cross-linguistic variation. A rule-based theory will typically contain very different rules for each language, with

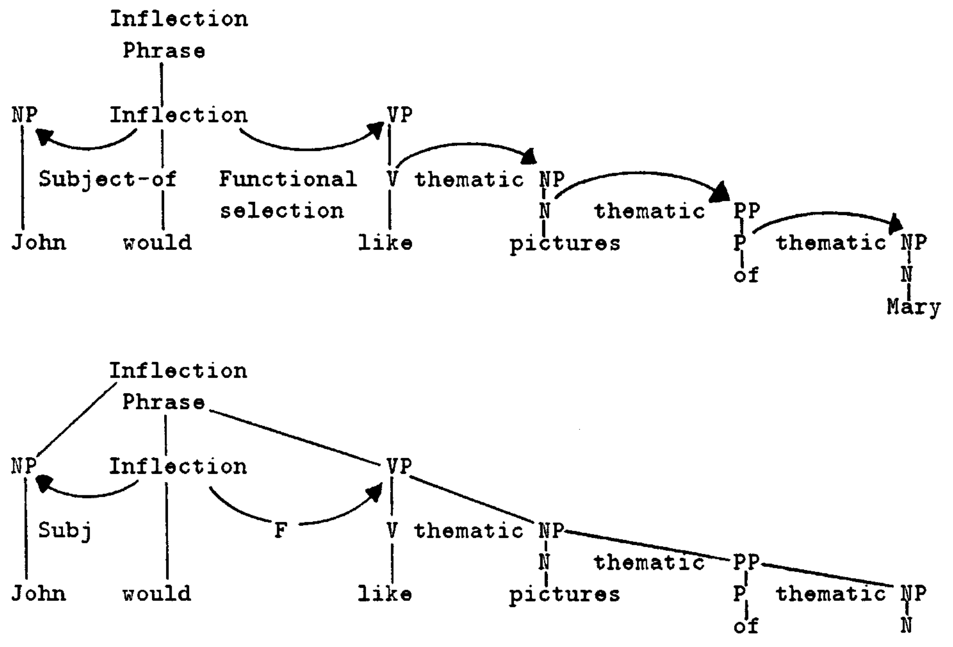


Figure 1: For a principle-based parser, phrase structure may be derivative.

no obvious connection between the two. As an example, consider any context-free rule based translation system, for instance, Slocum's Spanish-English translation program. Slocum's system would need an entirely different set of context-free rules for English-Italian translation, simply because there need be no constraint relating, say, Spanish and English, beyond saying simply that the rules for both are context-free. The rules have nothing else in common at all. In the best case, a principle-based system will confront cross-linguistic variation head-on, describing parametric variation *within* one set of principles that accounts for the full space of natural language possibilities. Translation systems based on principles consist of a large central core, common to all languages, plus variations sanctioned by the principles—not context-free rules that have nothing in common. We will look at a specific example in section 6.

- Rules breed uniformity, while principles tend not to. The reason is simple: with rules, the great temptation is to add one or more new rules to cover each new sentence example. Since it is usually possible to shoehorn any single natural language sentence into a context-free rule format, or, for that matter, any single rule format, rule systems tend to exhibit a certain uniformity. (Of course, this is being a bit too glib; we are talking here only of syntactic coverage.) Again we obtain large, hard to manage grammars. In contrast, principle-based systems try to assign distinct information-bearing tasks to each principle—for example, each kind of licensing has a special job of its own to do, while binding operates in a different domain, with its distinct principles and representations. This *information factoring* is central to the enterprise of principle-based grammatical description, and its resulting modularity lends itself naturally to the solution of cross-linguistic variation and, as pointed out in Barton, Berwick, and Ristad (1987), even to efficient parallel processing. (This of course does not *bar* a rule-based system from having many different kinds of rules—a case in point being the ID/LP system, in which immediate dominance rules are separated from linear precedence rules—it's just that things seem to run in the direction of uniformity precisely because so much attention is paid to string set coverage. It also does not rule out the possibility of compiling principles into highly reflexive rule systems, a point to which we'll return in section 2.)

  It should be noted here that the advantage of such knowledge factoring is recognized even in rule-based systems. Context-free systems often use one kind of principle-based device: it's often remarked that we can describe natural languages "as if" they consist of context-free skeletons plus feature augmentations imposed on top of these; hence the current interest in such CF-augmented systems in one variety or another.

Taking that point to its logical conclusion, we ought to view possible surface sentences built up in a kind of layer-cake fashion, as the *linear superpositioning* of several autonomous modules, each with its own constraints. This has an obvious direct interpretation in so-called logic-based formalisms.[6] Figure 2 illustrates. Indeed, *wh* movement is often imposed on top of whatever basic machinery builds context-free skeletons (LFG, the Marcus parser, and Abney's parser described below all agree on this point). From another perspective this is not so surprising: after all, most would agree to partitioning the representational vocabulary at least into syntactic and semantic components; all the principle-based idea does is push that idea to its limit. Plainly, there are many ways that this decomposition might be done, and the rest of this paper will explore some of them. But for now we will simply note that this "layer cake" idea has borne enormous fruit in other areas of linguistics. Consider autosegmental theory: in some versions of the theory, obeying the so-called "no-crossing" convention, a surface phonological form is built by superimposing several autonomous constraint levels.

In the remainder of this paper I would like to take up the computational points that each of these rule-based/principle-based distinctions implies, describing in passing several currently implemented principle-based parsing systems.

Section 2 first sets out a general framework in which to describe principle-based parsers and probes what it means to say that a parser *directly* implements a grammar or whether a grammar is *compiled*. These notions require a more careful treatment than they have usually received; most accounts are too weak or too strong. Section 2 explores a specific formal language theory example that brings this question into sharp relief, and proposes a particular way of looking at how grammars can be implemented as parsers. Section 2 goes on to propose a particular way of looking at how grammars might be "used" by language processors. In particular, we would like to reconcile various psycholinguistic facts about the automaticity or fluency of language processing (as noted by L. Frazier (1986) in an important and interesting recent paper) with a principle-based approach.

Our chief aim here will be to exhibit a variety of ways of "compiling" principles into a parser, drawn from implemented systems, and to examine just how compiling affects parsing. It appears that there are roughly three basic kinds of compiling that have been used in implemented principle-based systems, two *internal* to representational levels and one applying *across* representational levels: (1) parameter fixing within a representational level (e.g., the language is head first or head final); (2) substitution of values within a

[6] "Nearly autonomous" would be a better word here, since it's plain that certain constraint modules depend on others; for example, bounding theory applies only to chains or structured elements otherwise licensed by $\overline{\text{X}}$ theory.

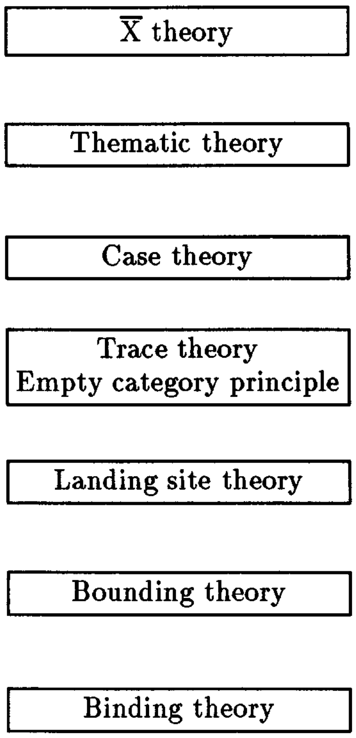


Figure 2: Nearly autonomous principle modules build up well-formedness conditions for surface sentences.

particular linguistic level (e.g., expanding an $\overline{X}$ skeleton by substituting lexical values for the X as Prep, Noun, Verb, Adjective); (3) substitution across representational levels, by folding together declarative constraints (e.g., multiplying out the various "landing sites" for empty categories would be an instance of substitution across the D-structure and S-structure levels).

Evidently, each of these kinds of compilation is no more than the sort of partial evaluation one often finds in Prolog programs, and one might take the notion of partial substitution (as Shieber does in his remarks (this volume), as one way to systematically examine the compilation relation and the space of possible "compiled" parsers. This is a useful approach, as far as it goes, since the notion of substitution and partial evaluation is well understood in the logic domain, and therefore the notion of "systematically related to" is direct. In section 2 we'll briefly examine three distinct logic-as-grammar approaches to principle-based parsing.

However, mere partial evaluation may not go far enough. If the notion of compilation is broadened to include the sophisticated things we expect from smart compilers, this amounts roughly to a re-axiomatization of the original system by a clever programmer, and any simple notion of "systematic" breaks down. The compilation need not even be automizable. Indeed, it's hard to see how the problem is conceptually different from that of automatic programming generally—a problem that computer science has yet to crack. Of course, we may still be able to prove that our compiler works faithfully, using the usual armamentarium of correctness proofs available to computer science, but this may well be possible only on a case-by-case basis. Put another way, unless we tie our hands rather severely, using Horn clauses or propositional formulas rather than Lisp or English to write down our description of a grammar doesn't really help tell us what the space of compiled systems is, even though the partial evaluation notion can give us some inkling as to where to start looking.

There have been two basic results from *practical* studies of principle compilation. First, in some compiled parsers, notably Dorr's (see section 6, psycholinguistic effects do not (need not) cluster along the fault lines of the principle module. If true, then psycholinguistic results indicating a different clustering than that dictated by a principle theory need not undermine that theory. Second, in several principle-based parsers where partial evaluation has been explored, this limited kind of partial evaluation compilation at first seems to speed up parsing, but then slows it down, if *too much* compilation is done. Here, "too much" compilation amounts to folding together all declarative principles of the theory, across all representational levels.[7] It's easy to see why this effect might occur: essentially,

[7] In general, this effect would *not* show up within a single level like, say, S-structure; it would not generally show up, for example, if the partially evaluated constraints included all surface structure constraints like

without any kind of substitution or partial evaluation, then the parser winds up building structural descriptions that have no chance of corresponding to any well-formed input. On the other hand, if too much partial evaluation is done across representational modules, then the system builds up a stock of multiplied-out structural descriptions (theorems) that again mostly don't apply to any particular sentence. Specifically, if the declarative system consists of $m_1 + m_2 \ldots m_n$ independent constraint modules, then in the worst case a fully substituted compiled system might have to sort through the cross-product of the module constraints, rather than checking the constraints additively.

The remainder of this paper will explore working principle-based parsers. Section 3 describes free-word order parsing for Warlpiri, comparing rule-based and principle-based approaches in the context of a parser implemented by my student M. Kashket (1986). Section 4 shows how rule-based and principle-based parsers deal differently with so-called ungrammatical sentences.

Section 5 covers two principle-based parser designs, showing how to avoid explicit (context-free) phrase structure rules and indicating the range of algorithm possibilities that a principle-based approach allows: Abney's licensing model and the direct use of $\overline{\mathrm{X}}$ rules in a deterministic parser. These examples will serve as a springboard for a discussion of how principles might be carved up in a parsing system. We will argue that there are three basic principle-based component clusters: (1) topological relations (like adjacency and hierarchy, often taken in context-free systems to mean simply general phrase structure rules, but in principle-based systems to mean $\overline{\mathrm{X}}$ topology); (2) lexical relations (like subcategorization and case marking), simply overlaid on top of or projected through the basic configurational relations (along the layer-cake or "autosegmental" model); and (3) binding relations (like *wh* movement and anaphor-antecedent relations generally), again overlaid on top of basic licensed structures.[8]

Section 6 turns to language translation, comparing a Spanish-English principle-based design implemented by my student B. Dorr with a standard context-free rule-based approach. Dorr's parser uses a bipartite division by grouping principle cluster (1) separately from (2) and (3): she exploits the topological information in an underspecified $\overline{\mathrm{X}}$ theory

Case theory, $\overline{\mathrm{X}}$ theory (without movement), and Government. It would show up if the constraints from a truly independent representational level were folded in, for example, if logical form constraints (scope binding ambiguities) were included.

[8]Berwick and Weinberg (1984) propose a two-stage parsing division that distinguishes principle clusters (1) and (2) from (3). Frazier's (1986) recent discussion on principle-based parsing and its psycholinguistic implications was is also relevant here. Frazier brings up many of the same points that I will discuss below, and I will draw on some of her psycholinguistic insights. In particular, she suggests a tripartite division of principles grounded on the fundamental notion of a "chain." While I have substantial disagreements with some of the specific points Frazier raises her remarks are quite intriguing and again show how a shift to a principle-based view leads to quite different ideas about the information structures required for parsing.

to drive an Earley-type parser that is co-routined (operates in tandem) with GB government constraints, idiosyncratic subcategorization, and the like. Whenever the skeletal $\overline{\text{X}}$ parse can no longer be extended, the GB constraint system is called to weed out underconstrained parsers or propose parsing extensions. Dorr's parser shows that principles may be clustered in a variety of different ways.

Finally, section 7 concludes with some brief speculative remarks about the computational power of modular, principle-based systems, along with possibilities for speedup using parallel computation.

2 A framework for principle-based parsers

The first question we turn to is this: What does it mean to have a principle-based parser at all? Intuitively, the answer is clear: It means that grammatical principles are *accessed* and *used* in parsing. Unfortunately, what it means to use a principle is unclear; further, it's even less clear what it means to *directly* use a grammatical principles. This point is important because principle-based parsers are usually assumed to be direct, and therefore evidence that parsing is somehow indirect may be cited as a count against principle-based parsing.

In this section we would like to examine what it means to implement a principle-based parser, either directly or indirectly.

- First, we'll show that the notion of "direct" use of grammatical principles demands clarification, and offer a formal language theory example that helps to bring this issue into sharp relief. We'll also examine how logic-based grammars give us three distinct notions of how principles might be used in parsing: directly; via metagrammatical interpretation; and via compilation. (Section 2.1)[9]
- Second, we'll define principle-based parsers in terms of the information clusters they use—whether they group $\overline{X}$ theory together with case theory or not, for example. We'll use this taxonomy to group various principle-based parsing proposals. We'll also look in more detail at one possible account of compilation, based on the logical notions of substitution or partial evaluation. (Section 2.2)
- Third, we'll examine the psycholinguistic arguments put forth by Frazier (1986) claiming that principle-based parsers don't look like good models for human sentence processing, whereas parsers that directly access and use ordinary phrase structure rules (that don't contain detailed lexical information) do fit the psycholinguistic facts. (Section 2.3)

What it means to use a principle or rule is complex in the case of grammars because several different senses of "use" are relevant. Grammars themselves play many functional roles: grammars can be used for language acquisition, or for sentence analysis or sentence production. Logically speaking then, representations of grammars could be used in one of these domains but not in another. For example, one could propose that grammatical universals are used for language acquisition but not for parsing. This was the essence of Fodor,

[9]I owe this observation to Shieber's comments (this volume), though in fact this tripartite distinction is quite standard in the logic programming literature; see, for example, Bowen and Kowalski (1982).

Bever, and Garrett's (1974) idea that grammars are used for learning, but then only their surface effects (surface word order cues such as the appearance of "that" before embedded complements) are consulted for parsing. To take an extreme version of this proposal, suppose the acquired grammar generated only a finite number of sentences. Then one could, in principle, first learn this grammar and then generate all possible sentences. One could then throw away the grammar and parse by consulting just the table of sentences. Still, one would properly say that the grammar was "used," since grammatical representations were used to carry out the construction of the sentence table. More realistically, an LR parser uses (finite-state) tables that are systematically constructed from an underlying unambiguous context-free grammar. As is well known from the compiler literature, such a parser can even handle ambiguous context-free grammars if one adds external oracles that resolve ambiguities that arise during the parse and show up as multiple entries in the table construction. In this case, the augmented parser is even more indirectly related to its grammar, though it is still systematically related.

This example shows that the relationship of "use" could be an indirect one, with "use in parsing" decoupled, in an explanatory sense, from "use in acquisition." Grammars can be represented in many different ways, and any of these could be used to generate the required sentence table. From the standpoint of the sentence analyzer, any of these grammars would be the same.

Such a possibility suggests that the attribution of principles or rules really does just follow from "best explanation": we say that people follow rules or access representations because in so doing we gain the best account we have of language acquisition and use. This is a completely unremarkable position but for some reason seems to have stirred wide controversy in the cognitive sciences. To see how unremarkable it is, consider a similar problem in S. Ullman's theory of the recovery of structure from visual motion. A key element in the theory is what Ullman (1979) calls the "rigidity assumption": any set of elements undergoing a two-dimensional transformation that has a unique interpretation as a rigid body moving in space is so interpreted. Suppose we take this principle as the "best explanation" we have of how people compute the structure of an object from viewing successive snapshots of it in motion. Then we can ask whether the rigidity principle is "used." Evidently, the answer is yes, even though we have no idea what its realization might be. Note that no one supposes that the Rigidity Principle is literally enscribed in the brain. Presumably, there is some physical basis for the RP itself, where the RP is not literally expressed. But this does not detract from the RP idealization, which successfully explains the operation of the visual system. The visual system apparently acts as if it used the Rigidity Principle.

It seems then that there's at least a strong and a weak sense of how a system might "use" a set of principles. In a strong sense, the principles (or in fact rules) might be literally represented as such, individually, and used directly online as data structures for parsing. For example, this is roughly how the rules of a context-free grammar might be used in, say, Earley's algorithm (though the analogy is imperfect because as a matter of course the literal rules are never encoded as such but are usually broken down into a more efficient tabular form). In a weaker sense, principles might be *interpreted*. This could admit varying degrees of partial evaluation: for example, as suggestion in the introduction, one could fold together constraints on $\overline{\text{X}}$ structures to get a set of partially instantiated trees with defined heads, rather than just underdetermined $\overline{\text{X}}$ skeletons. While it's clear what this comes to in the case of simple substitution of values for variables and partial evaluation, it's not so clear what this means when we admit *any* kind of transformation on the original set of principles.

To see if we can distinguish between these two cases we shall look at a particular artificial language where the direct use of a system is particularly clear, as a litmus test to dispose of some proposed definitions. We will then turn to three ways of looking at the same problem using logic grammars.

2.1 The notion of "direct" use of principles or rules

What does it mean to use a grammatical rule or principle? Many definitions have been offered, but the notion of implementation remains unclear, because we do not know what counts as an implementation, or what we are allowed to do to build our parser. For example, suppose, (for whatever accidental reason) that the sentences of a language contain only an even number of words. Is this a fact that the parser is allowed to consult in order to figure out how to process a sentence? Should parsers be allowed to count? If so, would this still be a direct implementation of a grammar? Consider, for example, word frequency effects, or Fodor-Bever-Garrett type heuristics ("take N–V boundaries to mark a sentence"). Suppose we can derive reliable surface cues from our principle theory; can these be used in a principle-based parser? Should we demand that surface cues be universal or language particular? Many questions arise; unfortunately, we don't have answers to most of them.

In order to maintain our intuitions about the connections between grammars and language processors, it seems clear that not just *any* fact of language counts as a direct implementation of a principle-based (or rule-based) parser. We use a particularly clearcut case from formal language theory below to drive home this point. Our moral will be that principles are used *directly* as long as they obey the representational encapsulations provided by the grammatical theory; as soon as we carry out substitutions or transformations that destroy the original representational levels described by the grammar, then the principles are being used *indirectly*.[10]

2.1.1 A formal language theory example

Let's turn first to the formal language example. Our case study centers on two example parsers modeled after an idea in Aho and Ullman (1972:272). Both parsers output same structural descriptions—they are strongly equivalent. But the first parser follows grammar rules directly, the second indirectly. The first accesses—"uses"—the rules of its corresponding context-free grammar, abiding representational integrity of those rules. The second uses the grammar derivatively by computing some (global) property of the grammar and then accessing that derived conclusion.

The language is a^+b. The grammar generating this language is given as this (with rules numbered for later reference):

(1) $S \rightarrow AC$ (2) $A \rightarrow AB$
(3) $A \rightarrow a$ (4) $B \rightarrow a$
(5) $C \rightarrow b$

For example, the sequence of steps used to generate the string *aaab* by a rightmost derivation in the grammar would be:[11]

1. S
2. AC (via rule 1)
3. Ab (via rule 5)
4. ABb (via rule 2)
5. Aab (via rule 4)
6. $ABab$ (via rule 2)
7. $Aaab$ (via rule 4)
8. $aaab$ (via rule 3)

[10]Note that this account would allow a parser to make *finer* distinctions than the grammar.
[11]Recall that a *rightmost* derivation expands the rightmost nonterminal at each step.

For our purposes here, a suitable output description of the parse of *aaab* would simply be the string of symbols 1524243, corresponding to the application of the rules of the grammar in their proper order, or the reverse order 3424251. An abstract description of the parsing problem for this sentence, then, is that the machine is given the string *aaab* and must output either string of numbers.

How should this be done? The first parser we shall describe will, in a clear sense, access and use the rules (1)–(5) directly. At all times its memory will hold exactly those tokens and strings of tokens admitted by the grammar in its derivation of the sentence in question. No other computation, no other "reasoning" will be permitted.[12]

The second parser will work differently, although it will recover exactly the same representation as the first.[13] The second machine does not consult a representation of the grammar at all, but rather relies on a fortuitous connection between the tokens of the language generated by the grammar and the underlying grammar.[14] In fact, for other context-free languages and grammar, e.g., $aa^+b + aa^+c$, Aho and Ullman show that a "natural" deterministic parser may not even exist.

The first parser operates by carrying out the rightmost derivation described earlier, but in reverse, using the "standard" simulation of a grammatical derivation via a pushdown automaton. This gives a so-called "canonical" LR (*L*eft-to right scan, *R*ight-most derivation) parse. The device is a simple bottom-up parser that uses a single pushdown stack and two simple parsing actions: (i) *push* a nonterminal symbol of the grammar on the stack and (ii) *replace* a sequence of symbols on the stack (corresponding to the righthand side of some rule expansion) with the lefthand side of that grammar rule.

Crucially, each action manipulates just lefthand and righthand sides of grammatical rules. We add an output action that is associated with each *replace* action that simply prints the corresponding rule number. In addition, at each step in the parse (=after each parser action) strings of tokens on the stack must be *isomorphic* to the corresponding line in the rightmost derivation, up to the point where terminal elements are encountered. A parsing action is defined by a stack string, input token pair, where the stack string may

[12]There is still admittedly a sense in which even this first parser will be indirect: namely, it will use a stack, just as the second parser does. Now, one might argue that this already involves auxiliary, non-linguistically sanctioned data structures, and hence is an indirect implementation. We might be able to dodge this by building a tree an using that instead of a stack. But we will sidestep this point here; there is still clear sense in which the second parser pays even less attention to the individual rules of the grammar.

[13]Thus we see that in certain cases the notion of "strong equivalence" is in fact too weak for some purposes: we cannot discriminate between these two strongly equivalent, but otherwise quite different systems.

[14]Note that even here the grammar is "used," namely to derive the original theorem, but once so used, the grammar may be discarded. The connection to language learning should be plain.

consist of several tokens (with an upper bound fixed in advance).

The operating rules of the parser are as follows. The machine starts with its pushdown stack empty and scanning the leftmost token of the input sentence. Besides the rules listed below, all other combinations of stack and input symbols lead to rejection of the input sentence.

1. If the leftmost input token is an a, and the top of stack holds the end of stack symbol, then push A on the stack, output the number 3, and read the next input token.
2. If the leftmost input token is an a, and there is an A on the top of the stack, then push B on the stack, output the number 4, and read the next input token.
3. If a B is on the top of the stack, and a A is the second token on the stack, then replace these with the token A, and output the number 2.
4. If the leftmost input token is a b and A is on the top of the stack, then push C on the stack, output the number 5, and read the next input token.
5. If a C is on the top of the stack, and A is the second token on the stack, then replace these with the token S and output the number 1.
6. If S is on the top of the stack and the second token on the stack is the end of stack marker, and there are no more input tokens, then accept the sentence and halt.

The sequence of parsing operations for analyzing *aaab* would run as follows. The important point to stress is that the successive "snapshots" of the stack configuration plus the remaining input duplicate (in reverse) the rightmost derivation. This correspondence is exact. For each line in a derivation, there is a corresponding parser configuration, and vice-versa. More precisely, there is a 1-1 correspondence between strings of machine configuration tokens and derivation line string tokens.

$$\begin{array}{cccc} & & \boxed{A} & \\ \boxed{\bullet} & \text{push } A & \boxed{\bullet} & \text{push } B \\ \text{1: Input: } aaab & \text{print } 3 \rightarrow & \text{2: Input: } aab & \text{output } 4 \rightarrow \end{array}$$

$$\begin{array}{cccc} \boxed{B} & & & \\ \boxed{A} & & \boxed{A} & \\ \boxed{\bullet} & \text{replace } BA & \boxed{\bullet} & \text{push } B \\ \text{3: Input: } ab & \text{print } 2 \rightarrow & \text{4: Input: } ab & \text{output } 4 \rightarrow \end{array}$$

$$\begin{array}{cccc} \boxed{B} & & & \\ \boxed{A} & & \boxed{A} & \\ \boxed{\bullet} & \text{replace } BA & \boxed{\bullet} & \text{push } C \\ \text{5: Input: } b & \text{print } 2 \rightarrow & \text{6: Input: } b & \text{print } 5 \rightarrow \end{array}$$

$$\begin{array}{cccc} \boxed{C} & & & \\ \boxed{A} & & \boxed{S} & \\ \boxed{\bullet} & \text{replace } CA & \boxed{\bullet} & \\ \text{7: Input:}\emptyset & \text{print } 1 \rightarrow & \text{8: Input:}\emptyset & \text{accept and halt.} \end{array}$$

Now let us look at the second parser that does the same job. This machine will *not* consult the token strings of rightmost derivations at all. Instead it will use a fortuitous property of this grammar that relates derivations to its surface language: we can use the a's to "count" how many occurrences of the $A \rightarrow AB$; $B \rightarrow a$ productions there have been. Here are the second parser's rules:

1. If the leftmost token of the input is an a, then push a onto the stack and read the next input token.
2. If the leftmost token of the input is a b, then push b onto the stack and read the next input token.
3. If the top of the stack is a b, and there are no more tokens in the input, then replace b with the empty string (this "pops" b off the stack) and output the numbers 1, 5.
4. If the top of the stack is an a, the next stack symbol is not the end of stack symbol, and there are no more tokens in the input, then replace a with the empty string and output the numbers 2, 4.

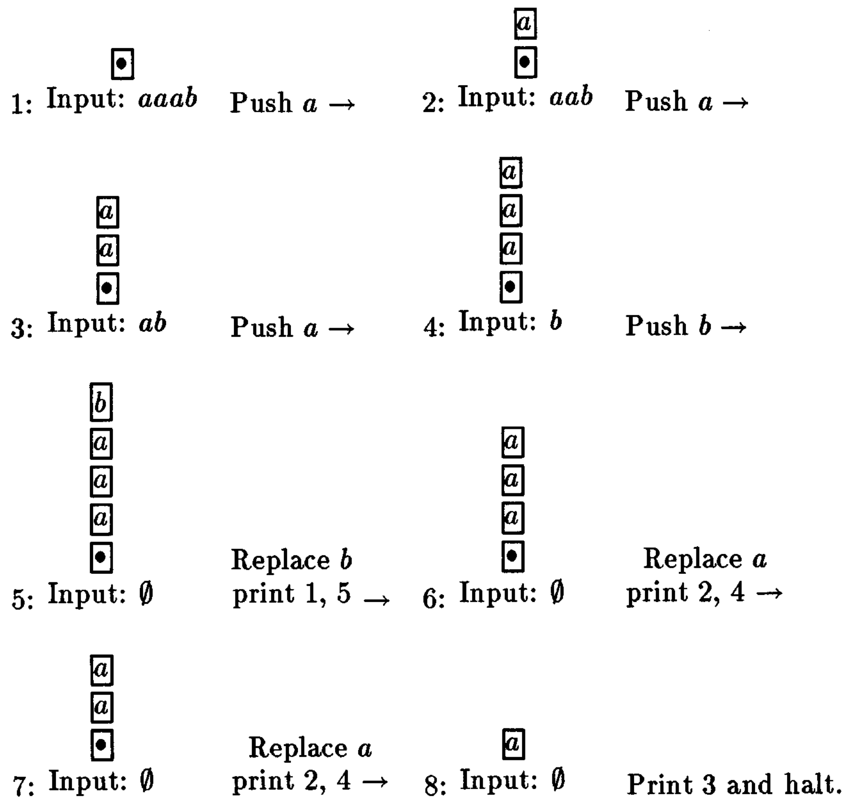


Figure 3: An indirect parser need not refer to a grammar at all.

5. If the top of stack is an *a* and the next stack symbol is the end of stack symbol, then output the number 3 and halt, accepting the sentence.

Note that this second machine does not mention the righthand or lefthand sides of grammar rules at all. Still, it is easy to see that this parser will output precisely the same structural descriptions as the first. And it uses exactly the same number of elementary steps; see figure 3.

Let's make this contrast explicit. The first parser actually refers to the grammar's literal constituent parts in order to decide what move to make. For example, if the first parser sees the unit AB on the stack, corresponding the righthand side of its rule (2), then it replaces that part with the lefthand side of rule (2). The form of the rules is directly implicated in the operation of the first machine.

In contrast, the second parser "uses" the grammar—indirectly, to determine just what derivation number to output. Unlike the "direct" parser, it does not refer to any of the units of the grammar—the left or righthand side components of rules. But it still uses facts about the language—indirect facts, but facts nonetheless. One would be hard pressed to say that the parser actually follows the grammar in parsing the language.

To see what the alternatives might be, let us first note that for the first sort of parser we can carry out the conversion to a parser mechanically. No matter what the rules of the grammar, there is a straightforward relationship between the right-and lefthand sides of grammar rules and the *push* and *replace* operations. Putting aside questions of the ambiguity of derivations, for every rule of the form $A \rightarrow w$, w a string of terminal tokens, the machine pushes an A onto its stack whenever w appears in an input sentence. Similarly, if there is a rule of the form $A \rightarrow \phi$, ϕ a string of nonterminals, then whenever ϕ appears on the stack then the machine can replace it with the symbol A. This means that the transparent relationship of the parser to the grammar will be invariant under counterfactual changes in the form of the grammar. Adding or subtracting a rule will not significantly alter the connection between grammar and parsing operations.

This property of conversion under counterfactual changes to original grammar does not hold for the second parser: here, adding just a single new rule to the original grammar will destroy the relatively fragile counting property that the parser exploited.

For example, consider the language consisting of just triples of a's, using the rules (1) S$\rightarrow$aaSa and (2) S$\rightarrow$ λ. A "derived" analyzer for this language could just count the number of a's and divide by 3, then output that number of 1's followed by a 2. If we now replace rule (2) with $S \rightarrow a$, then this parsing strategy now longer holds. Indeed, the "divide by 3" algorithm has all the properties of a parsing heuristic suggested by Fodor, Bever, and Garrett:

1. A parsing heuristic is ad hoc, in the sense that it may not work for all grammars.
2. In other words, a parsing heuristic is language particular, perhaps based on language-particular surface word cues.

This is suggestive, but it's not clear that this is a sufficient property to distinguish the two cases, however, since it's possible to take a context-free grammar—an LR(k) grammar—that has a systematically associated parser, and then destroy the LR(k)-ness by adding or subtracting one rule. But this means that an indirect system will generally lack explanatory power, because it cannot, in general, hold up under counterfactual conditions. This is the potentially great weakness of a "derived" machine (when in fact the surface behavior is derivable from other principles).[15]

[15]This distinction differs from other proposals that attempt to sort out the explicit/implicit rule spectrum.

In any case, the example deflates some other proposed criteria for "direct" parser embeddings. Consider take Pylyshyn's complexity measure (1984): Pylyshyn suggests that one process is computationally equivalent to another if one process executes in time n while the second operates in time kn, k a constant fixed in advance. But this is plainly not a sufficient condition, since both parsers 1 and 2 take the same number of steps; both run in linear time in the length of input sentences. We can refine Pylyshyn's criterion from a *linear* time constraint to a *realtime* constraint. A machine is *realtime* if it takes at most a bounded number of computational steps between each time it reads an input token, with the bound fixed in advance. Parser 1 is realtime, but parser 2 is not, since it takes a linear

---

Dennett (1983) proposes three categories of rules: explicit, implicit, and tacit. For Dennett, *explicit* rules are those that are literally enscribed, accessed, and causally figure in the operation of a process. In clear cases of rule-following behavior, a representation of a rule is empowered by its *very shape* as a causative agent of the process in which it participates. To use Matthews's (1984) terminology, in this case we say that a representation ... "figures in the etiology of ... behavior." The system actually consults a representation of a rule in order to figure out what to do next. A simple example: one might produce simple sentences by consulting the literal enscription, "Sentence → Noun Phrase Verb Phrase."

In contrast, Dennett defines *implicit* knowledge just as that word would suggest: as knowledge that can be implied by knowledge that is stored explicitly. For example, my knowledge of whether a certain sentence English 100 words long is grammatical or not is presumably derivable from my knowledge of the grammar of English, hence is implicit knowledge.

Finally, for Dennett, *tacit* knowledge is that simply "built into" the operation of a process—the brute "know how" that cuts off any Rylean infinite regress. On the familiar terrain of Turing machines, this will correspond to whatever machinery it is that knows how to interpret the instructions such as "move the tape head one square to the left"; for an ordinary computer, this will be its built-in ability to decode an instruction (in binary form) and respond appropriately (updating its program counter, etc.).

On this account "tacit knowledge" comes out sounding very much like physical reductionism pure and simple. Consider the computer's ability to execute instructions. Even this description introduces entities that do not exist except under our attribution: for instance, physically speaking, there need be no literal "program counter" at all in a computer. Of course, some machines *do* possess a physically distinct piece of hardware that serves as a program counter, but this is by no means necessary; other microprocessors simply use a distinguished memory address, or register. Entities like *program counter* and *instruction register* serve to regularize our theories of the behavior of the machine; without assuming them, the machine's step-by-step operation may be quite mysterious. Indeed, it would seem that the description of the machine in terms of program counter, instruction register, and the like, is on a par with theoretical descriptions in other sciences, and, by extension, on a par with theories of cognitive processes. At bottom, all behavior is presumably "tacit" in Dennett's sense, since it is physical regularities and primitives that fix behavioral capabilities. It is an uninteresting fact that we can replace any program description—any set of rules, if you will—with an equivalent version that is simply "wired" to behave in just the way prescribed by the rules, because the description in terms of voltages and wires is not the crucial one. The true theory of the machine—the counterfactual supporting description—uses the terms "program counter" and so forth, simply because the regularities of the computer mesh with this theoretical vocabulary.[16] Similarly for grammars. We introduce the terms S, NP, $\overline{\text{X}}$ level, and constraints like the Projection Principle, Subjacency, and so forth, for the exactly the same reason, so as to best account for the facts of natural grammars.

number of steps after all tokens of the input are read.

In its strongest form then, we could say that a *direct* implementation is allowed to use only the bare axioms of the grammar—the individual grammatical principles themselves. Weakening this somewhat, an indirect—though still principle-based—parser could be obtained by allowing any variable substitution or axiom system partial evaluation while still respecting the representational levels in the original grammar. For example, in a context-free grammar, if we had the preterminal chain rules NP→Noun; Noun→Name, then we ought to be able to substitute *Name* for *Noun*.[17]

To take another simple example of "compilation", consider the three-part definition of *properly_governs* from section one. We can "fold" this definition together into one clause by simple substitution to eliminate the intermediate predicate *c_command* and save some other steps:

$$\begin{aligned} \mathit{properly\_governs}(X,Y) \leftarrow\ & \mathit{maximal\_projection}(X,Y) \wedge \\ & \mathit{dominates}(Xmax,X) \wedge \\ & \mathit{least\_maximal\_projection}(Y,Xmax) \wedge \\ & \mathit{first\_branching\_node\_from}(X,BrNode) \wedge \\ & \mathit{dominates}(BrNode,Y) \wedge \\ & \neg\mathit{dominates}(X,Y) \wedge \\ & \neg\mathit{dominates}(Y,X). \end{aligned}$$

Note that this sort of partial evaluation still abides by the informational encapsulation of the original grammar even though it is a derived fact about the grammar. Or consider the usual LR table construction; here, a particular table entry collapses together into an equivalence class all lefthand sentence contexts that "lead to the same state." LR table entries reflect globally derived properties of their grammars, but still respect their grammars' original derivational structures. Put another way, the table entries and the parser's individual computation steps are transparently interpretable as grammatical derivation steps. But that's not true of our unnatural parser 2: here, the association between derivation steps and parser operations is only interpretable as such because we have externally

[17]Another simple example drawn from logic grammars—which incidentally provides one of the simplest possible examples of "compiling"—is how one deals with terminal words within a Horn clause translation of a phrase structure rule. For example, assuming that $A.U$ stands for a list with a single head element A and a tail U, then McCord (1987:318) initially writes the context-free rule NP→ Det Noun *that* VP as $\mathrm{np}(X,V) \leftarrow \mathrm{det}(X,Y) \wedge \mathrm{noun}(Y,Z) \wedge (Z = \mathit{that}.U) \wedge \mathrm{vp}(U,V)$. But, as he notes, we can just substitute for Z directly, obtaining the simpler expression: $\mathrm{np}(X,V) \leftarrow \mathrm{det}(X,Y) \wedge \mathrm{noun}(Y,\mathit{that}.U) \wedge \mathrm{vp}(U,V)$. Note that this substitution obeys the representational boundaries of the original principles (the single phrase structure rule).

supplied the interpretation relation; it is not mechanically connected to the grammar as with the LR tables.[18]

More generally, given a set of declarative constraints (an axiom system), a direct, principle-based parser could still allow variable substitution (as in the chain rule example above), or partial evaluation of those axioms. The parser would be an indirect, principle-based system if it used derived theorems that did not reflect the information encapsulation of the original system—e.g., if it used for parsing the fact that all sentences of a language began with *the*.

2.1.2 An application of the definition

Let's apply this test for direct–indirect principle/rule use to a real example, and then turn in the next subsection to a logical analysis of direct–indirect parsers.

Consider the Marcus parser, in particular how the Marcus parser analyzes passive sentences. The parser has a rule that inserts a trace into the position just after a verb, if that verb is marked passive; in turn, this marker is present just in those cases where the verb in question is passivizable and where passive morphology has been detected earlier. But this parsing rule does not *directly* use the grammatical principles of modern transformational theory (the Government-Binding theory). The rule says only to "insert a trace if the verb is passive." But there is no such "rule" in the GB theory. Rather, the possibility of this action is licensed (and forced, in English) by a constellation of other principles: the Projection Principle forces an NP to appear as a complement to a passivizable verb; the Head-Complement order of English puts this position just after the verb; passive morphology eliminates the possibility of case assignment by the verb to this NP, so it must move to Subject position, hence there must be an empty element that can survive the case filter in the position just after the verb. The Marcus rule is implied by these more basic principles, as a theorem is implied by an axiom system. Plainly this is a derived rule.

Matthews (1984) argues that the Marcus parser—crucially, as originally programmed—does not *directly* embed a transformational grammar. The rules that the Marcus parser accesses to guide its operation do not "look" like transformational rules. As we saw in the case of passive, there need be no 1-1 correspondence between Marcus parser rules and transformational rules. Pushed to its extreme, in fact, the Marcus parser could operate using just Fodor, Bever, and Garrett "heuristic strategies," in the manner of Parser 2 described earlier. (One can write Marcus parser rules that "implement" the rules of Parser

[18] Another possibility here it to introduce the usual sort of complexity metrics in order to quantify derivational complexity and so measure the "distance" between axioms and derived predicates. This would be a worthwhile alternative to pursue, but is beyond the scope of this survey.

2, for example.)

On the other hand, it is also possible to write other Marcus parser rule sets in the manner of Parser 1, so that the rules *do* directly reflect the representations of transformational rules, at least according to certain formalizations of transformational rules. Let us take as an example the Lasnik and Kupin (1977) formalization. One can show that Marcus-type rules can be written that mirror the Lasnik and Kupin definition of transformations. A full account would take us far afield, so here we shall simply describe just one example. In the Lasnik and Kupin framework, the structural changes (SC) described by a transformation must be written in one of the following forms:

$$(A_1A_2A_3, i/j)$$

$$(A_1A_2A_3, (i + j_r))$$

$$(A_1A_2A_3, (i + j_l))$$

where

$$1 \leq i \leq n, i \neq j$$

These changes correspond to movement and right- or left- adjunction. In addition, deletion and insertion transformations are defined as follows:

$$(A_1A_2, \emptyset/j)$$

$$(A_1A_2, b/j), 1 \leq j \leq n$$

where b is a member of "some language-specific set of 'insertable elements' " (Lasnik and Kupin 1977:180). The elements A_i are associated with particular phrase structure analysis fragments called *sub-reduced phrase markers.* Very roughly, these fragments correspond to tree fragments as specified by a Marcus parser's rule triggering pattern. Of course, since the Marcus parser is mapping from a surface sentence to an annotated surface structure instead of mapping from a D-structure to an annotated surface string, the parsing rules must "undo" the structural changes themselves.

Note that there can be at most three terms in a Lasnik and Kupin structural change. They also note that there are *implied* variables between each term: that is, there may be some arbitrary tree material between each A_i. Thus there are five elements possible in a structural change: three explicit items, and two implicit "don't care" positions. This property is easily replicated in the Marcus parser design by using the five possible elements of a rule pattern, the three buffer cell elements and the two active node stack items.

For example, consider the (inverse) of *wh* movement. In the Lasnik and Kupin framework, this may be stated as:

$$(\text{COMP WH}, (2/1)),$$

that is, the *wh* element replaces the COMP. The inverse operation must move a *wh* item from a COMP position to a "landing site." The corresponding Marcus-type rule would simply be stated as,

> cyclic node: COMP+wh;
>
> action: insert trace into the buffer

This structural change is, of course, restricted to operate according to a whole battery of conditions: strict cyclicity, subjacency and tensed S conditions, and so forth. These conditions are not part of transformational rules themselves—they do not appear in the *form* of the rules. Rather, they are universal conditions that apply, as it were, to the operation of transformational derivations generally. Again, the operation of our modified Marcus machine is similar: the parser is built so that it obeys the conditions of strict cyclicity, subjacency, and so forth; these conditions are not present in rule statements. Most of the constraints that prevent overgeneration are actually encoded in these general principles. In addition, unlike the problem of mapping from D- to S-structure, the Marcus parser needs some way to guarantee that it is building an analysis that conforms to D-structure principles. This is assumed by Lasnik and Kupin; the transformational analysis *starts* with a well-formed D-structure. To capture these constraints, the Marcus parser must add another set of constraints, not present in the the transformational rules or the t-rule operating principles themselves. These constraints are associated with lexical entries dictating the required argument structure of verbs and the Marcus parser's "packet system" that fixes canonical base phrase structure order. This knowledge is lumped together in the original Marcus parser but may be separated out so that grammar rules themselves state only Lasnik and Kupin type operations.[19]

2.2 Principle clustering, compilation, and logic grammars

There is plenty of evidence for automaticity in fluent parsing—recall the Fodor, Bever, and Garrett (1974) heuristics that allowed for such statements as "take the pair N–V as a sentence unless there is a surface sign of an embedding." Frazier (1986) provides other recent psycholinguistic evidence pointing to such automaticity in parsing, which is after all a quite natural assumption.

All of these examples, anecdotal or otherwise, point in the direction of compilation. Now, as the introduction pointed out, "compilation" can mean many things, from simple

[19] A more extensive demonstration of the connection between the Lasnik and Kupin formalism and the Marcus parser must await more time and space.

variable substitution, to partial evaluation, to re-axiomatization—the construction of some complicated, specific program that carries out some arbitrary mapping of the original principle set to a new one. We have already seen how one can multiply-out the principles in a logic grammar by simple substitution and partial evaluation, as in the three-part definition of *properly_governs*. In this section, we will examine this issue in more detail, taking a closer look at how logic grammars might help us delimit the space of principle-based parsers.

First note that what one regards as theorems is grammar relative. One can always build a new system that includes old axioms plus the derived theorems as axioms. However, it is easy to exclude this case simply by requiring that our axiom set be primitive in the sense that no axiom be derivable from any subset of the other axioms. We might also replace old axioms with derived theorems, but then we must be careful that the proposed representational units are just as explanatorily adequate as before. These restrictions have a particularly clear interpretation in the case of rewrite systems. Consider the following example. Suppose we had this grammar:

$$\begin{array}{ll} S \rightarrow S_1 & S_1 \rightarrow S_2 a \\ S_2 \rightarrow abc & \end{array}$$

A direct parser must use the representational units as given by the grammar: the symbols S, S_1, S_2, and the push and replace parsing actions corresponding to each rules as described above. As before, we could construct a derived parser that uses the rule $S \rightarrow abca$ directly. Now, we could add the new derived rule $S \rightarrow abca$ to our old rules, and claim a direct implementation. But this would violate the condition that representational units be respected. Or, we could replace rules (1)–(3) with this single new rule. The new axiom would be independent, and our parser now direct. But we must be careful that this single rule is all that is required. Our grammatical representation now excludes the units S_1, S_2, and so forth. If such representational units are required on independent grounds, then we cannot exclude them. All this means is that our new system is subject to the usual demands of descriptive and explanatory adequacy.

Considering the results of the previous section then, we may distinguish at least three distinct notions of principle-based implementation, the first direct, and the second and third indirect:[20]

- *Direct* implementation: the principles are used for parsing as represented, without any intervening recoding. We may imagine the principles to be represented and

[20]The distinction is again spelled out by Shieber (this volume).

consulted as if they were written down directly inside the parser itself, and consulted on-line.

- *Indirect parsing by weak compilation*: the principles are preprocessed by an interpreter that works on-line, so that principles can be processed only on a principle-by-principle basis (in essence, a metagrammatical approach).
- *Indirect parsing by strong compilation*: the principles are preprocessed ("compiled") by a (perhaps arbitrary) computational procedure into a form that is actually used for parsing. For example, this is what happens in so-called LR parsing: a (context-free) grammar is processed to output a set of parsing tables that are actually used, and the grammar then may be discarded.[21] We may take this preprocessing to range all the way from systematic partial evaluation of the metagrammatical interpreter to its wholesale replacement (by whatever means at our disposal—e.g., a human programmer's inspection).

We return to the logic-based view below, but first cover some general points and some specific observations about actually implemented systems.

The compilation issue is an important and delicate one. As Shieber (1985:145) notes, in the case of systems with complex features, "compilation" in the sense of full expansion into atomic features, usable by the Earley algorithm, may be problematic: one may fail to obtain closure on the *predict* operation of the Earley algorithm or in the standard LR parsing algorithm, for example. Shieber suggests restricting feature expansion in the PREDICT step of the Earley algorithm, so that the parser may still operate.[22] In essence, this is the key to how principle-based parsers are designed: they explicitly cut-off theorem derivations beyond a certain point so as to get working parsing algorithms. The problem is that this leaves open just how compilation is to be done; just what are the reliable derived or clustered principles to use? Do they vary from language to language?[23]

In practice, implemented principle-based parsers most often use three kinds of simple compilation techniques. The first two operate *within* a given linguistic level of representation, like S-structure, and include parameter fixing (is the language head first or head final) and variable substitution (fleshing out an $\overline{X}$ skeleton tree by setting the Head to

[21]Note that the compilation procedure might result in an incomplete parser, that is, one that does not work on all inputs without some kind of external oracle; this what can happen when an ambiguous context-free grammar is compiled into LR parsing tables.

[22]This is done by lumping a possibly infinite domain of nonterminals into one of a finite number of equivalence classes, deemed relevant for parsing.

[23]Note that the choice of the restrictor function in Shieber's proposal is really left open to the grammar programmer; there are arbitrarily many possibilities.

the values Noun, Verb, Preposition, and so forth. The third commonly-used technique is partial evaluation: the basic axioms are multiplied out, generally *across* linguistic levels like D-structure and S-structure, to give, for example, all possible positions for movement within a single Sentence or within the range of two Sentences.

2.2.1 Compilation in a modular theory

Now, the many principles in a full-fledged modern grammatical theory will admit a variety of compilation possibilities, since in general these systems are "richly deductive"—that is, the axioms all tend to participate together in deriving surface sentence consequences. For example, if we have, say, seven distinct but connected axioms then as Shieber observes we can multiply out any 1 of these 7 possibilities, and apply any one of the remaining modules as a filter in any order. As we have suggested, all one is required to do is respect the proprietary *vocabulary* of their principles (their information structure) and certain logical dependencies (almost like dataflow dependencies) between the principle modules.[24] One can still combine them temporally in different ways and get a principle-based parser. It is by no means clear just which of these is "correct"; below we'll address the question of whether the logic-as-grammar view helps clarify matters.

What does seem common to almost all schemes is that $\overline{\text{X}}$ theory (either in an underspecified form or fully projected lexical items) serves as a basic template on which to rest the remaining parsing principles. In this respect, principle-based parsers agree with the commonly-held computational intuition that there's something important about a context-free base or skeleton for parsing natural languages. Only, a principle-based theory tells us *why* that's so: it's so because so many of the licensing conditions of case and thematic theory depend upon $\overline{\text{X}}$ configurations. We'll see below why a logic-based account gives us the same moral.

On the other hand, it's apparently not necessary to build $\overline{\text{X}}$ skeletons first, before filling in other licensing relationships. Consider Abney's parser. It works by analyzing two words at a time, determining whether one element case marks its neighbor or not. Only then does it project an $\overline{\text{X}}$ skeleton. We see then that we do not have a clear idea of the dataflow

[24]Crain and Fodor (1985) have carried out experiments, which, as they say, do rule out some possibilities: it looks like constraints are being consulted before the entire sentence is being processed; at the same time, it looks like information is sometimes accessed on an "as needed" basis. What one needs to do, as I'll emphasize repeatedly below, is carry out a much more detailed and careful dependency analysis of the modern theories, from GB to GPSG, which all possess quite subtle and not so subtle interrelationships between their modules. This would tell us something more detailed about which constraints need to be considered when.

dependencies in principle-based theories.[25]

Let us examine some actual examples to see what the range of possibilities for compiling principle-based parsers might be, sticking to models that build $\overline{\text{X}}$ skeletons first. We will defer the complications introduced by word frequency effects, intonational contours, and the like, although these certainly are other approaches worth investigating.[26] Figure 5 outlines some of these for a GB-based approach; it does not list all the required modules.

The paradigmatic example of an uncompiled principle-based system is direct, bottom-up $\overline{\text{X}}$ projection from lexical items. The idea is a simple one, and has been proposed by Berwick (1982); Pinker (1984); Abney (1986); Frazier (1986); Correa (1987) and many other researchers. In its clearest form, direct $\overline{\text{X}}$ projection has three steps:

1. Encounter a lexical item in the input;
2. Create a maximal projection based on the features of the item;
3. Assemble projected phrases (bottom-up) based on licensing from case marking, thematic theory, or whatever other principles one might import.

Note that there are many variations on just what lexical features might be projected, leading to different $\overline{\text{X}}$ parsers and different psycholinguistic predictions. Following Shieber's point about restriction, we might not want to project all features from the lexicon (indeed, the lexicon itself might be structured in a complicated way so as to force this choice). The possibilities range from the most underspecified to the least underspecified:

1. Project *nothing* about categorial features. This leads to a "pure" $\overline{\text{X}}$ skeleton parser.
2. Project just categorial features, but not detailed verb subcategorizations.
3. Project categorial features and verb subcategorizations. This can lead to Aspects-style phrase structure rules, as Frazier (1986) notes.

[25]There is at least a hint that whatever the derived or compiled relationships are, they seem to be grounded on binary relations. For example, there is no tripartite relation where a dative verb selects one NP argument, which in turn selects another NP argument; instead, selection is mediated just through the verb. To take another example, even parasitic gaps may be described as the intersection of two chains with the same head; chains are themselves composed of binary-related elements. If this observation is correct, then all derived facts about surface constructions used in compiled principle-based parsers might be restricted to binary relations in some fashion.

[26]Berwick (1982) shows how government and precedence parsing might be related, while Chomsky and Miller (1963) give the classic account of how intonational phrasing might be used to guide an initial skeletal parser that is only later refined. Note that in the Chomsky and Miller model the initial parser does not even build the same structures as those demanded by the competence grammar.

4. Project all features, including agreement features.

The most literally-minded principle-based $\overline{X}$ parser is the first. This model has been adopted by Kashket and (nearly) by Dorr (see sections 3 and 6)[27] Not unexpectedly, there seems to be some tradeoff here between parsing speed and use of pure $\overline{X}$ skeletons: if we use just bare $\overline{X}$ skeleton's Dorr's research shows that parsing is slowed down, because we have to wade through many dozens of illegitimate tree structures. To get around this problem, Dorr co-routines her parser with GB principles.

If we project topological structure and some modest amount of lexical information, we build up trees with NPs and VPs in them, but no information on subcategorization. Based on psycholinguistic evidence, Frazier (1986) proposes this as one possibility for what she dubs a non-principled based parser: one that would reflexively build VPs without looking at whether a verb is transitive or intransitive. Evidently, there is some evidence (from verb final languages like Dutch and from psycholinguistic experiments in English) that reflects this compilation choice. Frazier aims to "multiply out" the $\overline{X}$ choices for a particular language at least down to the level of N, V, A, and P.[28]

But this isn't necessary. If we used Dorr's parser, we could simply co-routine the tree-building routine with filtering constraints (figure 4). The parser works using any ordinary CF parsing algorithm—Earley's algorithm, CKY, a left-corner scheme (for English). Whenever the Earley module gets stuck at SCAN, PUSH, or POP time, the GB module is accessed to filter, extend, or refine the parse, then passing control back to the CF parser itself. Since this scheme works with relatively underspecified $\overline{X}$ skeletons (Dorr's) system, I see no reason why it should not work with NPs and VPs as well. In fact, Dorr (personal communication) suggests that this would speed up her working parser somewhat.

At the other extreme, we could attempt to project all lexical information into the $\overline{X}$ phrases, down to the level of agreement features (including slash features in a GPSG system). This is probably too extreme; note that many implemented parsers superimpose agreement (via registers or some other extragrammatical machinery) on top of phrase structure skeletons.

So much for the $\overline{X}$ component. How should the other principle modules be carved up? Clearly, there are some logical dependencies: for instance, since binding is computed in terms of c-command, we must first establish structural dependencies before proceeding to binding. It remains to establish these dataflow-like dependencies for all grammati-

[27]Dorr actually factors in some information about trace and adjunct possibilities.

[28]She also wants to multiply out into individual rules case theory and thematic theory, but it is hard to see exactly where these principles are used, so I will put this question aside for now and just consider the projection of lexical category information.

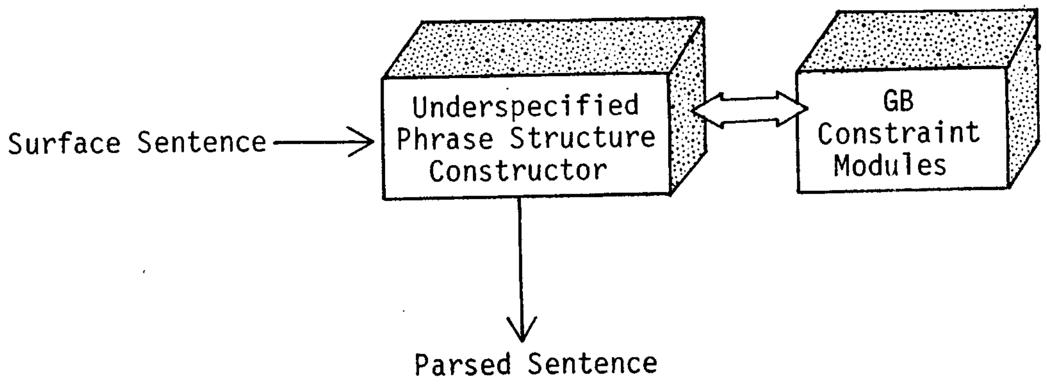


Figure 4: A tree-skeleton parser can be co-routined with other principles.

cal theories—as we pointed out, Abney's parser computes case relations before building tree structure—but a review of existing implemented principle-based parsers indicates the following compilation or clustering possibilities:

1. Compile out $\overline{\text{X}}$ principles plus just a few parameters from trace theory (choice of trace) to parse with underspecified $\overline{\text{X}}$ skeletons; the remaining principles are in separate modules and are accessed all separately on-line, with binding theory accessed after (in particular, subcategorization is accessed on-line). Example: Dorr's parser (see section 6).

2. Compile out an underspecified $\overline{\text{X}}$ parser that accesses subcategorization information held in common between lexical items. Example: Berwick and Barton (1985).

3. Compile out $\overline{\text{X}}$ principles, Case theory, and Thematic theory; this leads to old-style phrase structure rules, for example, VP$\rightarrow$V NP; with disjuncts allowed. Major category names are accessed and projected. Example: Frazier (1986).

4. Compile out $\overline{\text{X}}$ principles, case theory, subcategorization, some binding and bounding principles. Example: Sharp PROLOG GB parser (1985).

5. Compile out $\overline{\text{X}}$ principles, Case theory, Thematic theory, and subcategorization apart

from other principles. No existing principle-based system I know of does this exactly, though the Marcus parser comes close to this in an EST version.

6. Compile out subcategorization information apart from Case and Thematic theory, and apart from *wh*-movement, building $\overline{\text{X}}$ subtrees after case and thematic relationships are established. Example: Abney's parser (section 5).

7. Keep all principle modules distinct. Example: Kashket parser (see section 3).[29]

Each of these models lead to different kinds of "object" grammars; different sets of compiled rules. Each respects the *logical* division of principles, but organizes information flow somewhat differently. As Crain and Fodor (1985:126) have noted, this evident flexibility poses something of a problem for those who would like to test such models psycholinguistically (see also section 2.3 below): it is hard to establish that there are separate constraint boxes, because all the constraints could simply be applied simultaneously, thereby giving the effect of rule-like application. This (perhaps gloomy) point is basically well taken.

However, taking into account a wide range of evidence and implemented parsers that actually work well using modestly-multiplied out $\overline{\text{X}}$ skeletons (this includes fragmentary evidence about the use of intonational contours in parsing; the Freedman and Forster (1985) evidence what is effectively a two-stage parsing model; the success of Kashket and Dorr's parsing models; and the parallel-speedup possibilities for decoupled modular systems described in section 7), I think that we are beginning to discern the real advantages of modular, principle-based parsers. None of these factors, in and of itself, votes strongly for a modular system; but, taken together, the evidence is beginning to accumulate. Part of the problem is that no one has really take the time to even sketch out the space of "dataflow" possibilities for the information structures in GB, modern GPSG, or any other modern linguistic theory for that matter, leaving us mostly in the dark about what the psycholinguistic possibilities are.

Putting aside psycholinguistic issues however, an interesting result that emerges from Dorr's investigations of partial evaluation compilation, as well as from other researchers' work on similar issues (Johnson, 1987 forthcoming), is that too little or *too much* compilation slows parsing down.

Dorr's results for her particular way of multiplying out principle components (figure 2.2.1) shows that while a completely uncompiled system runs too slowly, because there are too many $\overline{\text{X}}$ possibilities to weed out, a completely multiplied-out system is also in the same boat, because its grammar size is too large. Evidently, there is an optimal balance,

[29]Other government-binding parsers, like Wherli's (1984) appear to lump together subcategorization information, case, and thematic theory, but not enough detail is available in published form to be sure.

as shown in the figure, but it's not clear how this varies from design to design or from language to language. This is one place where using a logic-based formalism would make it easier to systematically explore the effects of partial evaluation, as Shieber notes (this volume).

Johnson obtains similar results. When the constraints from all linguistic levels are folded together, so that there is just a single level of representation that is computed instead of D- S- and Logical Form constraints separately, then parsing time nearly doubles,[30] because the parser must check for a whole set of multiplicative possibilities: quantifier scope ambiguity with or without moved NP elements, at any possible landing site. If these constraints are left in their original, encapsulated form, then the parser can sort through a much smaller set of possibilities in an additive fashion: first check NP movement, and then check for quantifier scope possibilities.

2.2.2 Compilation in logic grammars

To sort out these possibilities, let's now reconsider how logic-based grammars tackle the problem of principle-based parsing. In section 1, we sketched what some of the declarative constraints for GB theory might look like. How would these be used for parsing?

A first approach would be simply to adopt the generate-and-test paradigm of logic programming. To do this properly, since the constraint filters of GB theory apply to trees or subtrees, we would have use the usual approach to record the structure of a proof as we go, by adding an extra dummy predicate (Green's trick); this will store the derivation tree. The declarative GB constraints would be applied to this derivation tree.

However, this method would be extraordinarily time consuming—essentially like the old analysis by synthesis approach first proposed for transformational grammars. It's easy to see why: Full sentences would be instantiated, and then checked for all possible constraints, but most of the time this would be wasteful. For example, the system would check for an ECP violation even in sentences where no movement rules had applied, and therefore no traces were present.

To see how to get around this problem, it is worthwhile to examine implemented principle-based GB parsers, such as Dorr's or Correa's (1987) (Correa's uses Prolog and so is particularly appropriate in this context). As noted earlier, these systems apply constraints as a parse tree is built up, subtree by subtree, and left to right. In Correa's system, this is done by passing up and constructing at appropriate mother nodes integer values that denote possible "chains" of NP or *wh* movement; this evaluation is done in a

[30]These are rough, hand-timed estimates.

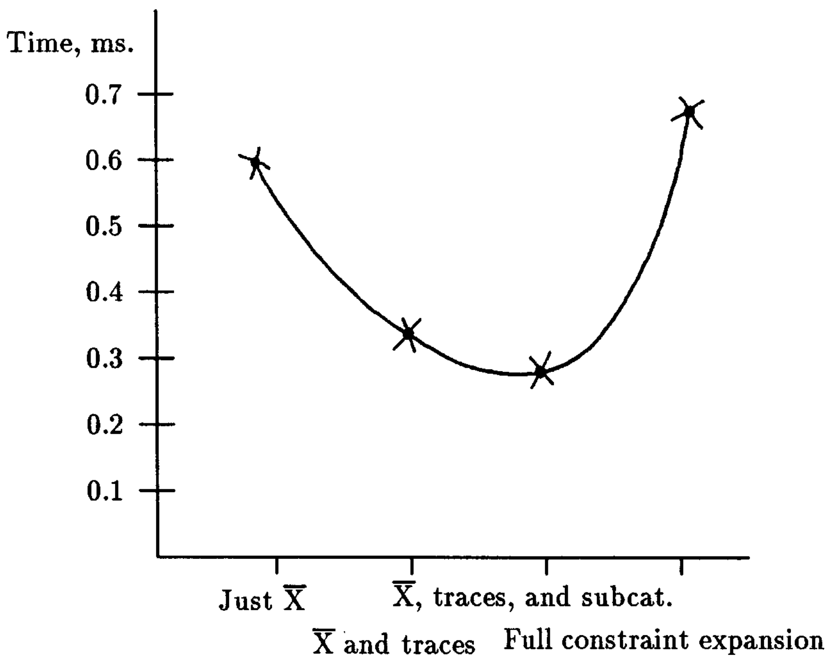


Figure 5: Too little or too much compilation slows down parsing in Dorr's principle-based parsing system. The Y-axis represents parsing time in tenths of a second for a single sentence, *Qué vio?*; times are similar for other sentences. The X-axis is not scaled but represents different compilation possibilities along an array from fewest constraints multiplied out to most constraints multiplied out.

syntax-directed way, by pseudo-semantic evaluation as each NP or S subtree is completed.[31]

This common approach gives us an important clue to how a logic-based GB parser could be made to work better. Instead of analysis-by-synthesis, one should apply constraints as the derivation tree is built up, thus weeding out bad cases as early as possible. One standard way to do this is to adopt a provability metainterpreter (Bowen and Kowalski 1982) so that each *step* in the deduction becomes accessible to control. If we do this, plus add extra variables to pass the constraint contexts around, much as GPSG uses its slash notation to percolate information about possible empty categories, then we now ought to be able to test constraints as we go, although it's not immediately obvious just what the new declarative constraints look like. I leave this as a suggestion to pursue in future research; it gives us a principle-based parser that may be "weakly compiled" by running the metainterpreter some number of steps, as Shieber suggests (this volume).

Finally, there is the possibility of "strong compilation": if we study exactly *when* constraints are applied, then, just as the Marcus parser contained an if-then rule to say exactly where a trace could be inserted, we ought to be able to see what constraints are applied at what steps in a derivation. For example, we know that the ECP will be applied only in trace-governed contexts, or that case is assigned only under government, so we can "factor in" this constraint to each rule that expands an NP to a trace by adding the ECP constraint clause (whatever that happens to be). While this may well obtain the most efficient possible parser, just as in the Marcus case, strong compilation may leave us with a nonsystematic relation between our original axiom system and our compiled one—in the limit, we're left to our own devices and programming cleverness to figure out just what the right compiled representation ought to be.[32] For example, Marcus used a special sort of buffer-plus-stack data structure; if that is in fact the processor that's most efficient, somehow the logic grammar compiler has got to output a device that simulates such a system. In general, this automation of data structure production seems quite difficult.

To summarize then: On the positive side, a logic grammar viewpoint certainly helps us sort out some of the compilation possibilities, if we take compilation to mean the relatively

[31]The use of such derivation chains to record movement history is remarkably similar—though with important variation—in theories as diverse as GPSG, lexical-functional grammar (LFG), and GB. This is the sense in which derivation structure is still crucial to syntactic theory, even though derivational history, in the sense of the order in which the mapping from one level to another is carried out, seems less crucial. It remains to formalize this notion of chain or path to see whether there are real differences between the various accounts. It is also important to note that this problem does not arise in simple-minded declarative GB reconstructions that axiomatize constraints only for single S's or one-level embedded S-structures, because then there are no chains to reconstruct. Thus the results from such simple analyses often belie the real complexity of a full declarative formulation of GB theory, or even of GPSG.

[32]Indeed, a glance at the Prolog compilers actually found in the literature will confirm this intuition; see, for example, the system of Dahl and McCord (1983), that handles conjunction.

simple sort of substitution operations that people have actually used so far. On this score, logic grammars do quite well, because we understand the systematicity of partial evaluation. On the other hand, for the most efficient designs, radically different data structures may be required, shedding the systematic relationship between principles and compiled systems due to outside programming intervention. If one is permitted to write a sophisticated compiler that can deduce special properties of the language in question and use them, then the beneficial systematicity between principles and object grammar breaks down; the compiler in question may be ad hoc because *any* recursive relation between principles and compiled system is allowed. In this case, the helpful word deduction becomes just a metaphor, stretched so thin as to lose any theoretical weight; if "deductions" encompass any sort of computations, then every parser is deductive in this sense and no special insight is gained. Finally, as we noted in the introduction, declarative principles may apply only to selected linguistic levels of representation—to surface structure and logical form, almost certainly, but not, so far as the empirical evidence would have it, to phonological form.

2.3 Reconciling principle-based parsers with psycholinguistic models

As described immediately above, there has been some debate as to whether one can use experimental results to figure out whether people access rules or modular constraint systems in online processing. On the one hand, researchers like Frazier, Clifton, and Randall (1983) and Freedman and Forster (1985) argue for a modular, constraint-based system, basically one in which structures are computed before constraints like Subjacency are applied. On the other hand, Crain and Fodor (1985) have countered with experiments claiming to reveal certain shortcomings in the original experiments, such that one wouldn't necessarily be able to conclude that constraints were being applied separately. I have neither the expertise nor the space here to delve into this dispute, so my aims will be much more narrow and modest: and that is to show that principle-based parsers are still, at least, compatible with recent psycholinguistic evidence presented by Frazier (1986). Naturally, this doesn't resolve the dispute about rules vs. constraints; as indicated above, we need a much deeper and more subtle analysis of the entire situation.

Turning now to these narrower points, Frazier (1986) has argued that principle-based parsing models make psycholinguistic predictions at odds with known psycholinguistic facts. Frazier presents two basic examples: (1) in Head final languages like Dutch the parser seems to blindly project NPs and do attachment, without first waiting to see what the verb is; this suggests that "pure" lexical projection is not carried out; (2) in some psycholinguistic experiments the human sentence processor apparently chooses transitive verbs blindly, without looking at lexical information. To comport with this evidence,

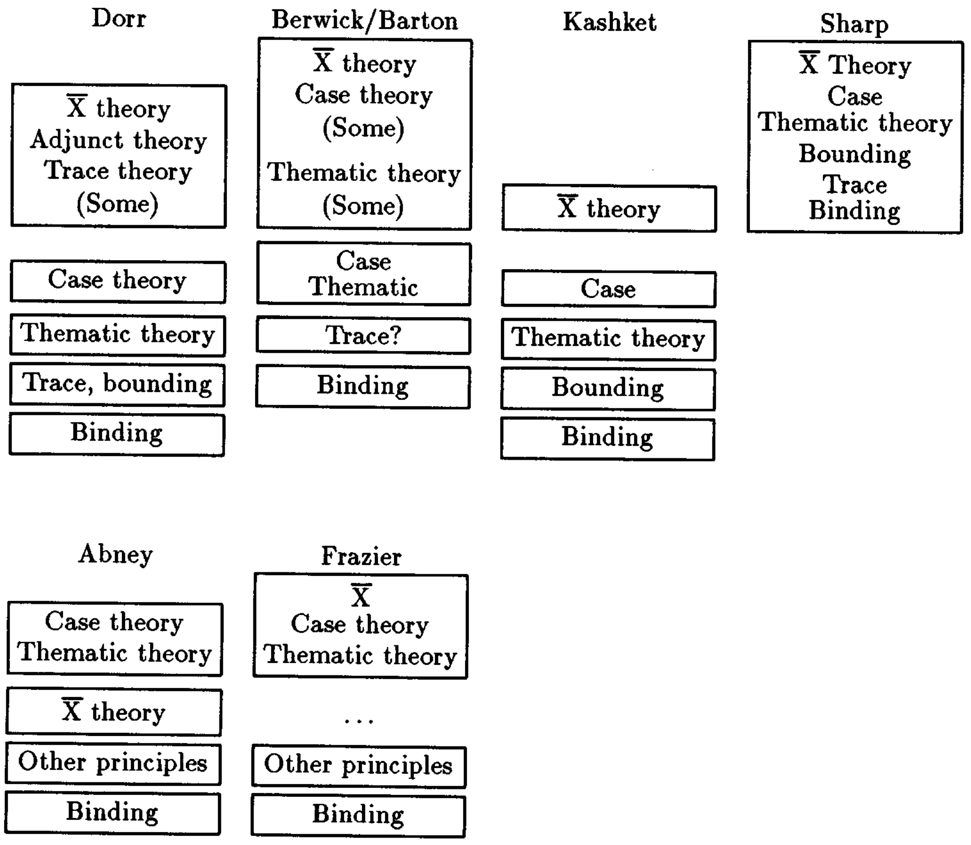


Figure 5: The same set of principles may be organized into distinct information flow structures, leading to different principle-based parsing designs.

Frazier argues for a slightly compiled-out rule system ($\overline{X}$ plus Case plus Thematic theory). Actually, she seems to be aiming for Aspects-style phrase structure rules than rules with full Case and Thematic constraints factored in, but that is irrelevant to the main point being considered here.

But neither of these psycholinguistic facts is really incompatible with a principle-based parser. Suppose we just use an underspecified $\overline{X}$ parser (perhaps constructing phrasal contours based on intonation, as has been suggested by some researchers). Whatever the details, we can certainly proceed with parsing, delaying access to subcategorization information until a verb is reached. This is exactly the way Dorr's parser works: the basic tree structure is built up before any other principles are consulted. Or, if one takes a more sophisticated view of what underspecification comes to, then Berwick and Barton's model (section 5) suggests another way around the problem. Both parsers are principle-based.

Kashket's parser design for Warlpiri also answers at least one of Frazier's objections. Kashket's parser is bottom-up in its basic operation, and projects maximal phrases from the lexicon. Yet it does not delay projection of phrases until the verb is encountered (since the verb may be encountered almost anywhere). Instead, each phrase subtree is constructed as it is found in the input.

Frazier argues that the absence of delays (as indicated by misanalyses that the Head would otherwise guard against) in processing Head final languages like Dutch argues against a direct projection of lexical item into $\overline{X}$ schemas. She also presents other intuitive and experimental evidence indicating that Dutch sentence processing preferences argue against Kashket's "direct projection" strategy. Frazier considers a Dutch (verb final) sentence like the following:

...dat	het	mesije	van	Holland	houdt
that	the	girl	from	Holland	liked

Frazier argues that when the preposition *van* is processed, it can only be projected to a PP and then attached to the preceding NP *the girl*, since the verb has not yet been seen. But evidently this is not the route taken; only if there is an additional PP, as in the sentence below, will the first PP be attached to the NP:

...dat	het	mesije	van	Holland	van	Spans	houdt
that	the	girl	from	Holland	from	Spain	liked

But all this assumes Frazier's minimal attachment strategy: that nodes are attached as soon as possible to the phrase structure tree being built. This is not a necessary constraint (though it is classically argued that it reduces memory load in some fashion). Suppose, in contrast, that one opts for a more "wait-and-see" approach, like Marcus'—and by this we mean simply a bottom-up processing strategy like Kashket's Kashket's processing approach for Warlpiri works well in this situation and is compatible with the Dutch processing evidence: it simply builds subtrees and waits until the main verb Head is projected. Any structural attachment difficulties that arise in the Dutch examples can now be dealt with. [33]

It remains to see just what psycholinguistic evidence can be brought to bear on the question of principle-based parsing. Certainly the gap between surface measurements and internal principles may be large. More precisely, we *don't* want to repeat the DTC mistakes all over again, and assume that logical divisions of principles must be temporally respected. On the other hand, Crain and Fodor's point about flexibility vitiating experimental power will then come back to haunt us. Given our current state of ignorance, I think the only way to settle this issue is to keep building parsers based on different designs, doing more natural history and exploring the space of possibilities. We also need to couple that with a more sophisticated formal analysis, drawing in some ideas from computer science, including those of distributed computing.

Again, a good example is Dorr's co-routined design. If no one had ever tried to build such a parser, we wouldn't know something about its unusual outward appearance: If one looks at the operation of this parser from the outside, then it looks like Case theory and Trace theory are being computed almost at the same time as $\overline{\mathrm{X}}$ skeletons, so one might conclude—erroneously—that these modules are all multiplied out together (thus *concretely* confirming Crain and Fodor's worry). In actuality they are kept distinct in this system. The problem is that we don't even know very much about what surface cues people actually use when they parse and how these are related to grammatical principles. Such cautionary tales suggest, unfortunately, that psycholinguistics may be even harder than it looks.

[33]This is not to say that there are difficulties in using Marcus' rule-based approach in a language like German. Indeed, our experience in writing a large Marcus-type system for German, as part of the Athena Foreign Language project at MIT, showed quite clearly that the verb-final character of the language caused problems, because important subcategorization information was not available.

3 Parsing free-word order languages

In this section I'll outline the different ways that a (context-free) rule system and a principle-based parser tackle Warlpiri, a well known free-word order language. I will argue that a principle-based parser does far better than a context-free rule system for such languages, and is even easily extended to English without modifying the parser machinery itself. Again, the key idea is to develop a more flexible representational vocabulary so that the possible surface constructions in a language may be easily described by parametric variations in principles, rather than by writing down a completely new and unrelated set of rules.

For concreteness, I will use the principle-based parser developed by my student M. Kashket, comparing it to context-free rules that parse the same language. I will also adopt his analysis of Warlpiri, in turn based on proposals by K. Hale and M. Laughren.

We will see that by using a vocabulary other than the *concatenation* and *hierarchy* blended together in context-free rules, namely, selection/case marking operating at the word level and case assignment principles operating at the phrasal level, we can easily account for the free-word order found in Warlpiri *as well as* that part of English that appears exhibits fixed-word order (Subject-Verb-Object) and those parts of English that are relatively free (Prepositional Phrase adjuncts). The key point is that the *only* difference between a parser for Warlpiri and one for English will be stated in terms of principled, parametric differences between the two languages that have to be stated anyway. One and the same parser will work for both.[34]

In contrast, since context-free rules possess only concatenation (linear position) to encode the more basic principles of case marking and case assignment, they ultimately fail to perspicuously describe the range of possibilities seen in natural languages. The result is that a rule-based parser for free-word order languages almost invariably writes out all possible word order sequences, leading to a corresponding increase in grammar size.[35]

3.1 Warlpiri and context-free parsing

To begin, let's consider some variations in a single Warlpiri sentence. (Hyphens are added to separate morphemes for readability):

[34]It was recently brought to my attention that Klavans (1982) proposes a principle-based division of LFG along roughly the same lines. This just underscores the advantages of principles over rules.

[35]We will take note of some attempts that have been made to bypass this problem, but we believe that these ultimately miss the essential point that the context-free rules are simply the wrong vocabulary for encoding the essential properties of Warlpiri, and, indeed, languages such as English.

Ngajulu-rlu	ka-rna-rla	punta-rni	kurdu-ku	karli
I	AUX	take	child	boomerang

I am taking the boomerang from the child

Kurdu-ku	ka-rna-rla	ngajulu-rlu	punta-rni	karli
child	AUX	I	take	boomerang

From the child I am taking the boomerang

Karli	ka-rna-rla	kurdu-ku	ngajulu-rlu	punta-rni
boomerang	AUX	child	I	take

It is the boomerang I am taking from the child

Plus 21 others ...

Although phrase order is free except for the rigid AUX-second position (as in German), phrasal variations lead to different emphasis in topic and focus, as the translations indicate. In contrast, morpheme order is fixed: at the level of words, Warlpiri is a Head-final language, with markers *rlu* (Ergative), *rni* (Tense, nonpast), and *ku* (Dative) appearing word final. (The absolutive case marker is null and so does not show up explicitly on *boomerang*. Also, there are basically just two lexical categories: Nouns and Verbs.)

How could we write a context-free rule system to describe these constructions? Let us consider several possibilities and discuss their deficiencies.

First, aiming at mere string coverage, we could explicitly write out all possible phrasal expansions. (Here, the tags *S* and *O* stand for Subject and Object and we ignore the AUX element):

S → NP-S NP-O V
S → NP-S V NP-O
S → NP-O NP-S V
S → NP-O V NP-S
S → V NP-S NP-O
S → V NP-O NP-S

Plainly, this is an unperspicuous grammar that also suffers from computational defects. By explicitly writing out the rules, we have missed the basic fact that the phrase order is free. To put the same point another way, the grammar would be almost as simple (almost as small) if we omitted the last rule S→ V NP-O NP-S. And the grammar is large, because it contains more rules, and will thus run more slowly using standard context-free parsing algorithms. Perhaps more importantly, on some accounts of semantic interpretation, we demand certain hierarchical structures that explicitly represents the domination of the object by the verb, with the NP Subject external to the VP. Thus a better context-free grammar would be something like this:

S	→	NP-S VP
VP	→	V NP-O

But as is well known, this sort of grammar cannot parse the sentence order V NP-S NP-O. To get over this hurdle, various proposals have been made: invisible VP nodes; movement rules; ID/LP grammars; and the like. What these rescue operations have in common is some way to break apart the linear phrasal concatenation forced on us by context-free rules; and indeed there is something essentially right in any such attempt to free us from the rule straightjacket.

One could resort to a change in algorithm in order to overcome this hurdle. One such proposal that has been made in the context of a functional unification grammar for Finnish (Karttunen and Kay 1985:298–305) is to say simply that one particular phrasal order is stored and the permutations that actually appear are generated on demand. Thus, the base grammar remains small. As Karttunen and Kay put it, "the opportunity is to work with a much smaller grammar by embodying the permutation property in the algorithm itself."

As we will see, in effect this is the approach adopted to parse Warlpiri via principles. There is a key distinction: in the Warlpiri system, the difference lies not with some special algorithm, but probably where it ought to lie: with the statement of the grammar of Warlpiri. In fact, *nothing special* need be said about the Warlpiri parsing algorithm at all; it does not have to embody some permutation procedure, except implicitly as allowed by the principles of the grammar. Further, the very same parser will work for English as well—crucially, as mentioned, the only changes that have to be stated are the *linguistic* differences between English and Warlpiri, which have to be stated anyway.

3.2 Japanese and context-free parsing

Free phrase order is not just a parsing problem in Warlpiri. The same problem arises in Japanese. Tenney (1986) reports on her experience in writing a context-free parser for Japanese (using a modified LR(k) parser that can return all possible parses, dubbed the

"Malone parser"):[36]

> The free word order of Japanese syntax presents a major problem for a rule-based parser such as the Malone parser. Each rule must express a possible fixed word order—of which there are a great many in Japanese. This means that coverage of even a small number of basic structures requires a large number of rules, reducing the overall efficiency of the parser. This problem was in part circumscribed by labeling several different kinds of constructions with one name ('ARG'), but this solution has reduced efficiency elsewhere in the system. For example, some extra machinery is required to distinguish ARG's that are required by verbal case frames from those which are not. (1986:2)

As one example of the extra machinery Tenney means, we can take a look at Tenney's rule for sentences with one overt argument. (The annotation below the rule is a Lisp function that runs at the time the LR parser is about to complete the right-hand side of a rule; in this case it checks that the sentence in fact has one overt argument).

```
S→ Smod? ARG Smod? verb
sent1-arg
```

The rule collapses all ARGs together, but they must then be distinguished later on, as Tenney makes clear in her discussion of this rule:

> This [rule] parses a sentence with one overt argument. A case frame is created, the verb entered as the predicate, and the argument entered as Nominative (*ga*) Dative (*ni*), Accusative (*o*), or Instrumental (*de*), depending on the casemarker of the argument and the case frame of the verb. (If the verb belongs to the verb class vst [stative verbs] the semantic specifications in the cas frame must determine which *ga*-marked ARG goes in which case slot.) If the ARG has the node type Sbar, it is entered into the case frame as having clause case. If there are ARG's left over after the case frame is filled, they are entered into the case frame as mod[ifiers] and labeled postpositional phrase.

[36]Of course, one might question whether Japanese is truly "nonconfigurational" in the same sense as Warlpiri. On the view presented here, this distinction between "configurational" and "nonconfigurational" languages is meaningless; rather, some languages allow freer phrase order than others (for example, the order of adjunct Prepositional Phrases in English is largely free). Whether Japanese is considered configurational or not, its free phrase order presents difficulties for a traditional context-free parser where such ordering is explicitly spelled out in advance.

If there is a topic-marked ARG, do not enter it in the case frame until all the other ARG's have been entered. The topic is entered in the case frame in the following way:

1. The rule only applies if the topic is the first ARG in the clause. Otherwise, reject the parse.
2. Call a function to check whether there are any empty case slots in the case frame. If there are, fill the topic ARG into that slot. If there are not, fill the topic ARG into the case frame as a mod[ifier].
3. Label the parent S with the feature Topic.

(1986:6)

It's evident that what we gain in being able to generate free phrasal order—by having just one type of node, ARG—we lose again when we have to distinguish ARGs. Thus, while this method is workable, it is unwieldy.

Tenney notes a second problem with context-free rule-based Japanese descriptions, for NPs. We cannot allow free NP recursion because Japanese NPs constrain what possible constructions there can be at each level of recursion:

- At level 1, simple nouns and adjective recursion is permitted (*yasui hon*);
- At level 2, NPs may be joined by conjunctions (*hon to zasshi*; *kuruma ya jitensha*);
- At level 3, NPs may be followed by nominal quantifiers (*gakusei igai*; *gakusei inai*; *gakusei dake*);
- At level 4, an NP may be followed by a particle (*gakoku kara*);
- At the ARG level, an NP is followed some general postposition marker (*gakusei wa*).

As Tenney notes (1986:29), "all of these constructions may appear in one NP (*binboo na gakusei to kanemochi-no roojin igai ni wa*) ... but the hierarchy of NP levels may not be violated." The upshot of these constraints is that a rule-based system must include five distinct NP types, NP1–NP4, plus the ARG NP itself. (It remains an open question how to handle these distinctions in a principle-based parser, but the difficulty with the rule-based approach is clear enough.)

Finally, Tenney notes her rule system eliminates hierarchical structure, yet that is probably needed in Japanese: "no word order is more basic than any other, yet some facts about how pronouns are construed with noun phrases within a sentence point to the idea that the word order Subject-Object-Verb is most basic in Japanese.[37] To build this into the rule system would increase the number of rules dramatically." (1986b:5)

3.3 The limitations of context-free rules

To summarize, for Japanese we see that context-free rules fall short in two respects, as they did for Warlpiri: on the one hand, in order to accommodate free-word order, they must "flatten" hierarchical structure, increase grammar size, and blur distinctions that are then postponed for later ("semantic") processing; on the other hand, if they try to accommodate the hierarchical structure demanded in Japanese (and, in some account, in Warlpiri), they cannot describe all sentences.

All of this suggests some underlying flaw in the context-free rule format itself. As mentioned, various extensions have been proposed to patch things up: one can change the parsing algorithm, incorporating permutation directly; one can introduce "invisible" VP nodes for Semitic VSO languages (Aoun 1979); one can propose "discontinuous" constituents, and so forth. In a certain sense, all of these proposals try to do the same thing: they try to break apart how context-free rules strictly tie phrasal hierarchical information to word precedence information.

This immediately suggests that a solution to the free-word order problem is to recast the context-free account of word order and phrase hierarchy in terms of a different vocabulary of autonomous, modular principles. The central idea is identical to Halle's (1986) view of autosegmental representations as a kind of "spiral notebook": the "surface" phonological form is like the spiral part of the notebook, resulting from the intersection or projection onto one plane of several independent, noninteracting "leaves": vowel sequences, consonants, rhythmic patterns, and the like. Figure 6 illustrates.

This "multidimensional" viewpoint is exactly what is implied by Aoun's invisible VP analysis to handle V-S-O order in Arabic. To make the comparison explicit, suppose we wrote the Verb–NP-Object connection in one plane, and the NP-Subject connection in another, disjoint plane. Then the VP can dominate the object NP in one plane, and be linked with the Subject NP in another, without any "crossing" of lines. See figure 7. Put another way, both NPs and VP can participate in domination relationships, without saying anything at all about precedence; precedence is established when the two planes are projected to a single phonological line. (Another key information processing feature of this proposal is that it attempts to reduce all grammatical relationships to adjacency:

[37] This evidence is drawn from Saito's Ph.D. thesis, MIT, 1985

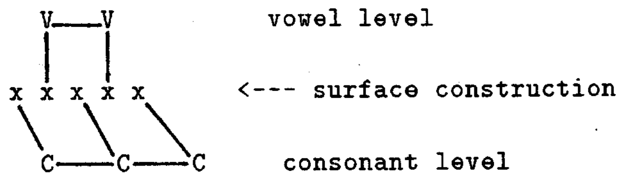


Figure 6: Halle's spiral notebook view of autonomous segmental components. They do not interact except to intersect at the surface level where a construction is ultimately observed.

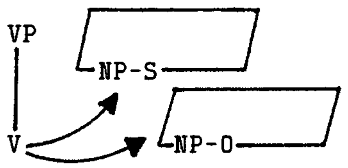


Figure 7: By dividing grammatical relations into separate "planes" we can describe a V-S-O language without crossing lines.

whatever looks like a "long distance" relation at the surface is in fact an adjacent one at a projected level of representation.)[38]

But note that this is just a *graphical* way of saying that we have broken apart the grammatical relationships between VP, Subject NP, and verb into autonomous, distinct modules that interact—in a word, we have broken up the context-free rules into principles. The whole idea of the principle-based approach is to abandon the straightjacket imposed by context-free rules to find a new vocabulary that will factor apart the constraints of word order and hierarchical phrases in such a way that they can operate independently, combining to produce the full range of free-word order to rigid-word order found in natural

[38]Verb-particle constructions exemplify the same phenomenon: if we "project" the verb-particle pair to a separate tier, then in fact they are adjacent even if not appearing so on the surface.

languages.[39]

Having stated what our goal is, let us see how we can discover the right principles to use and how we can get them to work directly for parsing Warlpiri (and English).

3.4 Parsing Warlpiri with principles

Kashket's key insight is to propose case marking and case assignment principles for Warlpiri and (at least) two autonomous representations at which these principles applies. One representation requires precedence-ordered trees and one does not, and this bifurcation allows us to account at the same time for the rigid morpheme order in Warlpiri along with its free word order.

- Morphemic: This level expresses, among other things, precedence relations (one morpheme precedes or follows another);
- Phrasal syntax: This level expresses hierarchical relations (one phrase dominates another). The phrasal elements bear no precedence relations to one another.

The claim, then, is that phrasal syntax really needs only hierarchical information, not precedence information (as is reflected quite generally in the order-free nature of many predicates such as c-command).[40]

Having split apart the representations, Kashket proposes to split apart case marking and case assignment along these very same representational fault lines. It is this division of principles that will allow us to capture the full range of free-word order—fixed-word order phenomena.

Case marking is taken to be an essentially morphophonemic (or phonological) process, hence one that logically ought to be represented at the level of morpheme structure. Therefore, case marking depends upon precedence information, since this is encoded at the morphophonemic level. As expected, it is directional, and operates completely locally, under adjacency.

In particular, in Warlpiri, case marking is to the left. This makes Warlpiri a Head final language at the morpheme level: case markers must appear at the end of words.[41] Let us review an example:

[39]Modularity also has suggestive advantages for parallel processing, as we will see in the last section of this paper.
[40]Other linguistic levels may require precedence; for example, if their is a "leftness" condition in logical form. We do not address this issue here.
[41]Some more subtle examples of Noun-Noun compounds are discussed below.

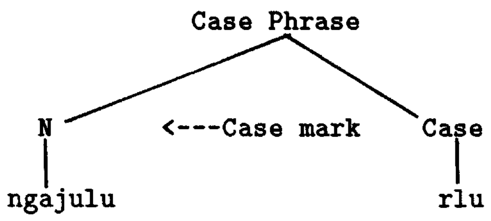


Figure 8: Directional case marking forces fixed morpheme order in Warlpiri.

In the word *ngajulu-rlu*, the ergative case marker *rlu* case marks *ngajulu* (*I*) to the left, so this word is well-formed; we may imagine it comprising a complete "case phrase" unit ready to be analyzed at the phrasal level, as indicated in figure 8.

Crucially, in Warlpiri verbs are *not* case markers. We shall see that this forces an essential difference between a so-called "configurational" language like English and "nonconfigurational" languages like English; the parser need only know that in English, a verb *is* a case marker, and that in Warlpiri, a verb *is not* a case marker.[42]

Case assignment is carried out at the phrasal level, under sisterhood, but with one crucial difference: since phrases do not even encode precedence information, case assignment cannot refer to order or adjacency at all and is nondirectional; this is what will allow Warlpiri phrases to be order free.

Let us consider another example to see how this works. Take the word sequence *ngajulu-rlu punta-rni karli* (*I took the boomerang*).[43] This is first parsed as three separate word-level units: *ngajulu-rlu* is a Noun-Case combination that is case marked as usual to form what Kashket calls a $\overline{\text{NC}}$ unit; *punta-rni* is a Verb-Tense combination that forms what Kashket calls a V unit; while *karli* has a null absolutive marker at its end, so is case-marked (as usual) to form a $\overline{\text{C}}$ phrase. Note that all three words are Head final and well-formed; if, for example, the tense marker had a noun to its left, then such a structure would be rejected.

The V morpheme unit is now projected into syntax under the usual $\overline{\text{X}}$ format: it contains a node, a $\overline{\text{V}}$ node, and a $\overline{\overline{\text{V}}}$ node. Under Kashket's model, the $\overline{\text{V}}$ node assigns

[42]The terms "configurational" and "nonconfigurational" are in this sense simply glosses for the deeper processes of case assignment and case marking.
[43]The AUX unit will be ignored for these examples.

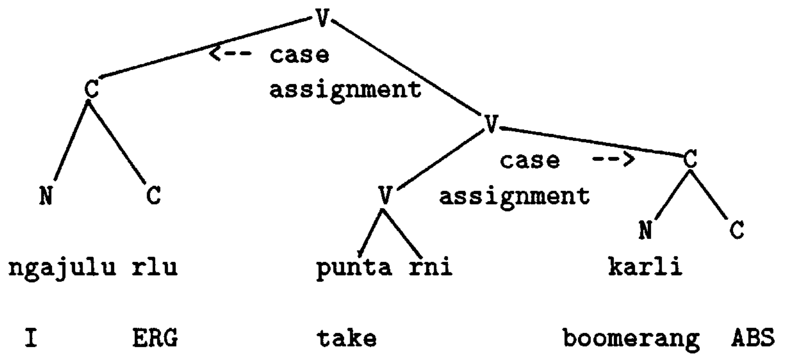


Figure 9: A Warlpiri syntactic structure. It is important to note that the order of subtrees is not encoded at this level, even though the picture must show some order

absolutive case (in either direction); since *karli* is marked for absolutive case, it receives this case no matter whether it is to the left or to the right of *punta-rni*. Similarly, $\overline{\overline{V}}$ case assigns ergative (either to the right or left) Finally, the association between thematic roles and cases is rigid in Warlpiri, so *karli* is identified as the THEME and *ngajulu-rlu* as AGENT:

AGENT	$\leftrightarrow$	ERGATIVE
THEME	$\leftrightarrow$	ABSOLUTIVE
SOURCE	$\leftrightarrow$	DATIVE

Figure 9 shows the resulting syntactic tree. Be sure not to be misled by the order of the subtrees shown in this structure; we must write them down in some order, but since case assignment does not obey adjacency and since ordering information is not even represented at the syntactic level, we could just have well interchanged the two $\overline{C}$ nodes. In other words, any of the six sequences *ngajulu-rlu punta-rni karli*, *punta-rni karli ngajulu-rlu*, *punta-rni karli ngajulu-rlu*, *punta-rni ngajulu-rlu karli*, *karli ngajulu-rlu punta-rni*, or *ngajulu-rlu karli punta-rni* would be correctly parsed.

Whether right or wrong, by splitting up case marking and case assignment principles—via adjacency and dominance—and using these principles in parsing, we can account for the difference between Warlpiri and English. In English, the verb *is* a case marker, so position matters: the verb case marks Subject and Object at the morpheme level where linear precedence is encoded. The result is that we get a rigid word order.

Note that English also exhibits some word order freedom: Prepositional Phrase NP Objects may appear in free order after a verb. Under the current account, that's because case assignment is carried out by the Preposition. Hence, at the phrasal level, there is no ordering constraint at all; all we say is that the VP dominates the PPs and NPs.

3.5 The Warlpiri parser accesses principles, not rules

With this overview behind us, we can now give the details of Kashket's parser, and cover some fine points omitted earlier.

The parser consists of two stages: a lexical parser that inputs morphemes and outputs an ordered forest of trees; and a phrasal parser that takes the lexical parser's output and attempts to produce a single hierarchical structure with unordered subtrees. No context-free rules or rules of any kind (in the usual sense) are accessed. A dictionary contains basic syntactic category information as well as *actions* (for a case or tense marker) that say what kind of element the case or tense marker selects (Noun or Verb), and, in the case of a verb, what arguments it case assigns. Figure 10 shows how the actual Warlpiri dictionary transparently encodes the case selection (marking), and assignment actions illustrated earlier; these properties are directly accessed by the parser. (Some of the data involving the auxiliary analysis will not be covered here. The *percolate* actions has to do with $\overline{\text{X}}$ projection of features.)

The lexical parser determines the well-formedness of words according to morphophonemic constraints. Basically, this stage operates on groups of two morphemes at a time. After the first morpheme is input no action can occur. The second input morpheme prompts word construction: the parser looks at the unit immediately to its left to see whether it may be combined (selected) by the case marker. For example, if the case marker is the tense element *rni*, and the unit to the left is not a verb, then the structure is ill-formed, and the two units remain detached; but if the unit to the left is a verb, then combination can occur and a V node produced, as described earlier. A dictionary is consulted here, as is typical. In addition, if a V projection (predicate) is being formed, the dictionary will supply case assignment actions to be associated with the projections of V, as appropriate. (Note that if there is a null case marker, as with the absolutive, then we assume that morphological analysis supplies a null second morpheme.)

As each word is completely constructed it is fed to the second stage, the phrasal parser.

```
RLA:        actions:  SELECT: (OBJECT ((AUXILIARY . SUBJECT)))
            data:     MORPHEME: RLA
                      NUMBER: SINGULAR
                      PERSON: 3
                      AUXILIARY: OBJECT
RNA:        actions:  SELECT: (SUBJECT ((AUXILIARY . BASE)))
            data:     MORPHEME: RNA
                      NUMBER: SINGULAR
                      PERSON 1
                      AUXILIARY: SUBJECT
KA:         actions:  SELECT: (AUXILIARY ((V . +) (N . -)))
            data:     MORPHEME: KA
                      TENSE: PRESENT
                      AUXILIARY: BASE
RNI:        actions:  SELECT (+ (( V. +) (N . -) (CONJUGATION . 2)))
                      ASSIGN: ABSOLUTIVE
            data:     MORPHEME: RNI
                      TENSE: NONPAST
                      TNS: +
PUNTA:      actions:
            data:     MORPHEME: PUNTA
                      THETA-ROLES: (AGENT THEME SOURCE)
                      CONJUGATION: 2
                      N: -
                      V: +
RLU:        actions:  SELECT: (ERGATIVE (( V. -)(N . +)))
                      MARK: ERGATIVE
            data:     MORPHEME: RLU
                      PERCOLATE: T
                      CASE: ERGATIVE
KU:         actions:  SELECT: (DATIVE ((V . -)((N . +)))
                      MARK: DATIVE
                      ASSIGN: DATIVE
            data:     MORPHEME: KU
                      PERCOLATE: T
                      CASE: DATIVE
NGAJULU     actions:
            data:     MORPHEME: NGAJULU
                      NUMBER: SINGULAR
                      PERSON: 1
                      N: +
                      V: -
KURDU:      actions:
            data:     MORPHEME: KURDU
                      N: +
                      V: -
KARLI:      actions:
            data:     MORPHEME: KARLI
                      N: +
                      V: -
```

Figure 10: The Warlpiri dictionary directly encodes $\overline{X}$ features, case marking, and case assignment.

This stage's job is simply to carry out case assignment, in effect "linking" arguments to any predicate and thereby licensing them. Recall that we assign case assignment actions to each V node projection, as retrieved from a dictionary. The phrasal parser will execute *all* applicable actions, globally, *until no more actions apply*, but, plainly, a case marked argument must be present before case assignment can take place. Note that the actions consider all possible directional case assignments, across all subphrases; this permits free phrasal order.

Finally, we note that the auxiliary is handled specially: its dictionary entry says that it takes a verb projection as an argument. (The aux-second constraint receives no explanation on this account.)

3.5.1 An example parse

An example should make this algorithm clearer. Consider an Object-Verb-Subject-Object sentence form, such as *kurdu-ku ka-rna-rla punta-rni ngajulu-rlu karli*. At the word level, nothing happens when the first morpheme *kurdu* is processed. The second morpheme, *ku*, adds a Dative case marker, and it selects the Noun to its left (a directional selection), forming complete word with Case as its head. The next morphemes comprise the Auxiliary and are projected as an A-phrase. Third, the verb "take" with its tense *rni* marker is seen; the tense selects the verb to its left, forming a V unit, which is passed to the phrasal parser. At the phrasal level, the tense marking itself is attached to the A(uxiliary) unit. Also at the phrasal level, the V is projected to a $\overline{\overline{V}}$. Now the morpheme *ngajulu* enters the input and is processed; it is noted as a Noun, but no actions can apply to it because it is not yet case marked. The next morpheme, *rlu*, combines to its left with the Noun to case mark it ergative and form a $\overline{C}$ phrase; this is passed to the phrasal parser. Figure 11 gives a snapshot of where things stand so far, where we have deliberately placed *ngajulu-rlu* to the left of the V projection node to indicate the order-free character of phrasal structure.

Now any actions attached to V projected nodes apply, if possible. Since there is an ergative case-marked argument, ergative case assignment applies, linking $\overline{C}$ to the V phrase. Similarly, the Dative assignment is possible and *kurdu-ku* is attached. See figure 12.

Since no more actions apply, *karli* enters the lexical parser; with its null absolute case marker, it is a well-formed word, and is passed as a $\overline{C}$ to the phrasal parser. Once again, a V projection action can now apply to this case marked argument, and assigns *karli* absolutive case (hence THEME of the sentence). figure 13 shows the final result, taken from a snapshot of the actual running parser. "PS" refers to the level of word structure, or Precedence Structure, while "SS" is surface phrase structure.

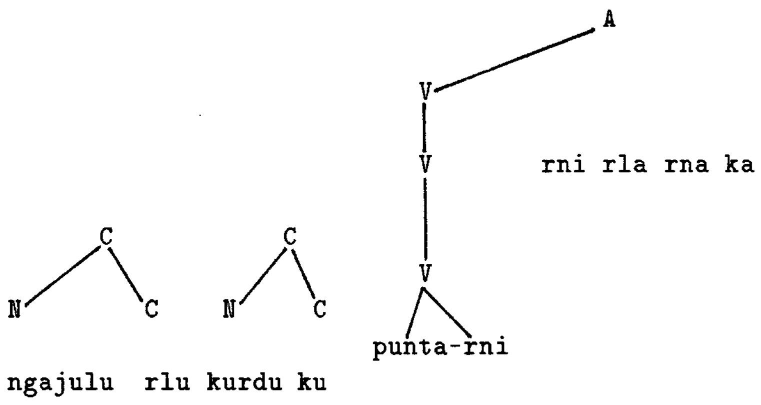


Figure 11: Starting to parse a Warlpiri sentence.

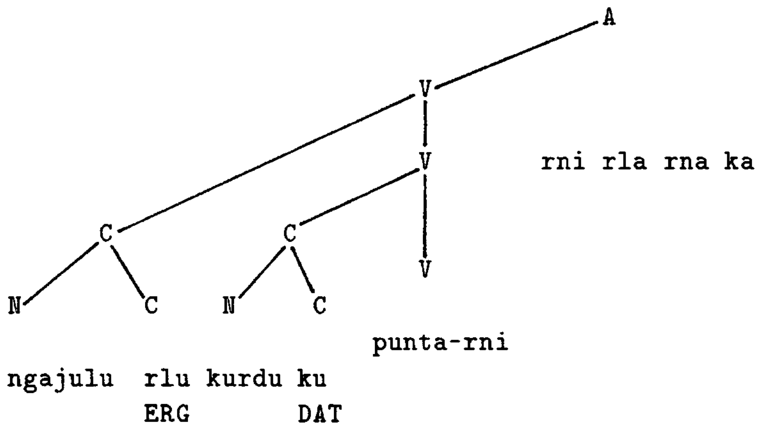
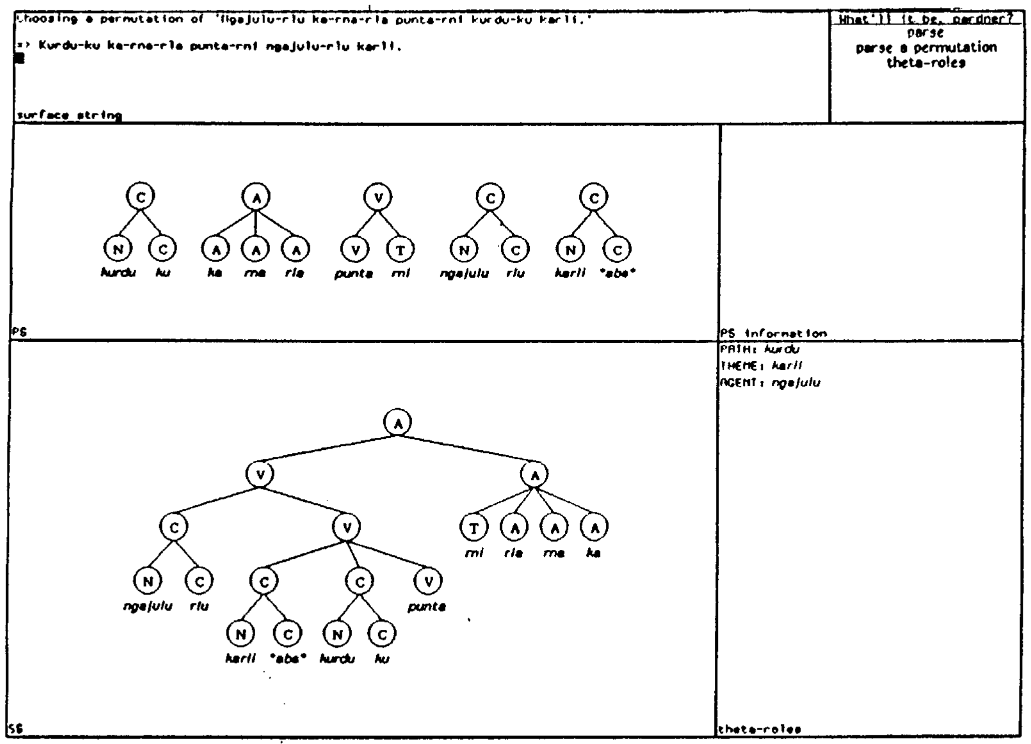


Figure 12: Linking the ergative subject and dative object in a Warlpiri sentence.



```
Choosing a permutation of 'Ngajulu-rlu ka-rna-rla punta-rni kurdu-ku karli.'

=> Kurdu-ku ka-rna-rla punta-rni ngajulu-rlu karli.

surface string
```

```
What'll it be, pardner?
parse
parse a permutation
theta-roles
```

```
PS information
PATH: kurdu
THEME: karli
AGENT: ngajulu
```

Figure 13: Linking in the Absolutive argument in the Warlpiri example sentence.

Kashket's implemented parser can handle a far wider range of constructions than shown here, including compound Nouns. These show a particularly interesting interaction between lexical parsing and phrasal parsing, and indicates the flexibility of a rule-less system. If there is a string of Nouns in the form:

Noun Noun Noun Noun ... Case

then there might be some ambiguity as to whether the unmarked Nouns get Absolutive case. But this does not occur; all the Nouns are transmitted the same case at the tail end of the phrase. The reason is that when Nouns appear in this kind of a group, they are all part of a *single* intonational phrase, and therefore can be phonologically recognized as a single unit. We may assume this preprocessing to take place prior or at the time of morpheme processing, as part of speech analysis. In contrast, if compound Nouns appear discontinuously, then they must all be case marked with the same marker, and again there is no parsing ambiguity. In this way, Kashket's system can quite easily accommodate additional information sources that are superimposed on his basic constraints.

Let us summarize why a principle-based approach succeeds where a context-free rule approach fails. Kashket's principled division into two distinct representations, a morphemic level obeying adjacency and linear precedence, and the phrasal obeying just dominance relations, forces rigid morpheme order and permits free phrase order. At the same time, the case marking/case assignment vocabulary lets us state the difference between languages like English and languages like Warlpiri in a straightforward grammatical way, without resorting to novel parsing procedures for either languages.[44]

In this regard, it is worthwhile to compare Kashket's system with the Kartunnen-Kay approach to free-word order parsing. Kashket's parser uses an unordered tree representation at the phrasal level. Thus, in effect he allows phrases to occur in any order, and they are so constructed on demand. This is precisely what Kartunnen and Kay sought to do with their free-word order parsing algorithm. The difference is that Kashket's permutations *follow* from a principled division in representation and case marking/case assignment, rather than from an arbitrary difference in algorithm. (Kay and Kartunnen could have just have easily *not* have introduced a permutation operation in their algorithm; again, this is because this kind of approach is construction-based, rather than principle-based.)

Indeed, Kashket's parsing algorithm for English and Warlpiri will look exactly the same. Nothing need be said other than what minimally must be said anyway about the

[44]A representational vocabulary grounded on the primitives of *precedes* and *dominates*—like that in the *Logical Structure of Linguistic Theory*—would be fully compatible with this approach; what has been added is the constraint that only certain levels resort to the *precedes* relation.

difference between the two languages: that verbs case mark in English, but not in Warlpiri. In this way, a uniform principle-based system can significantly advance our understanding of the free-word order/fixed-word order continuum, and show us how one kind of parser can handle many different kinds of languages.

4 Ungrammatical sentences and principle-based parsing

Perhaps one of the clearest ways to see the difference between rule- and principle-based parsing is to investigate how the two approaches treat "ungrammatical" or "ill-formed" sentences. There are two distinct issues here: one that has to do with what have traditionally been dubbed ungrammatical sentences, and another related issued that centers on the (largely GB based) distinction of core grammar vs. periphery—that is, sentences that are somehow central or basic to grammatical theory in some way vs. those that are stylistically marked. (The notion of stylistic variation is, of course, an empty one without some definitional substance behind it.) I'll take up both topics separately, though they are clearly closely related.

One important topic that I will *not* address in detail is the thorny one of what *people* do when they process ill-formed sentences. It's an important issue, because, as Freedman and Forster note, if their experimental results bifurcating grammar constraint violations like Subjacency vs. phrase structure and agreement violations are correct, then that indeed "presents problems for any theory that has not notion of degree of generability" (1985:125). A principle-based theory tackles the whole notion of degree of generability face on.

The problem here is that the psycholinguistic evidence is still under debate, and I for one don't really have anything to say about whether one experimental design is better than another. I will limit myself here to strictly the engineering differences between rules and principles. I would like to point out that almost all "working" parsers that have tried to deal with ill-formed sentences—from ATNs on—have tried to modularize their constraint systems, and not fold all rules together into a single system. The obvious example is agreement violations: since these are parceled out into register checks in an ATN, one can readily imagine the parse going through, but relaxing (separately) the register checks on agreement. (This would even fit what Crain and Fodor ultimately say about such cases, namely, that there's a longer matching time for such sentences because there's a search for a suitable grammatical alternative. But I wouldn't bet much money on this.)

4.1 Parsing ill-formed sentences: rules vs. principles

Parsing ill-formed sentences has always been a thorn in the side of rule-based systems. If we build a context-free grammar that can generate exactly the structural descriptions of a certain set of sentences, then how do we account for the perfectly human ability to assign structural descriptions to sentences not in this set? Indeed, grammaticality judgments would seem to demand that people build some kind of representation for ill-formed sentences, if only to be able to judge how they are ill-formed. There's no doubt, for example, that people *parse* sentences such as *What do I wonder John likes*, even though such sentences are difficult to interpret. Further, ungrammaticality can range from pure word salad—violations of basic phrasal order, as in *book the John put table on*—to thematic role violations, as in *John hit*—to binding violations.[45]

A principle-based approach accommodates the possibility of so-called ill-formed input naturally and directly in a way that a rule-based parser does not. Evidently, we might say that the parser assigns an interpretation to every string of sounds x—even if the sounds are Chinese for an English speaker.[46]

For a rule-based parser built from the start to handle just a distinguished set of strings, a language, this makes life difficult. What are we to do with strings not in the language? It is instructive to see how ill-formed input has been dealt with in rule-based systems. Essentially two approaches have been used: first, one can add new rules that parse the ungrammatical strings, while assigning scoring values to these parses; second, one can adopt a constraint relaxation model.

The first approach was adopted in the Diagram system (Robinson 1980). For example, the following DIAGRAM rule scores a sentence fragment such as *I saw boy* lower (giving it the feature"unlikely") than a fragment such as *I saw the boy*. (The Lisp-like style of the DIAGRAM rules has been replaced with a more English if-then format.)

[45]For an early attempt to describe the problem of ungrammatical sentences via a notion of *levels of grammaticality*, see *The Logical Structure of Linguistic Theory*. In effect, this approach makes no distinction between "grammatical" and "ungrammatical" sentences, and so does not even admit to a distinguished set of strings called a language. This work was of course written before the advent of formal language theory.

[46]We note simply in passing that this renders irrelevant the whole idea of weak generative capacity. Thus, while there is *something* correct in saying that natural languages are a lot like context-free languages, plus agreement augmentations and wh-movement, this statement is correct only in so far as it redescribes at a superficial level the more fundamental fact that natural languages follow $\bar{X}$ theory plus other constraints like case assignment, agreement conventions like the foot feature convention (if that is what one believes in), and so forth. Weak generative capacity is completely derivative, a side-effect of more fundamental principles.

```
If
            there is an NCOMP node
Then
            set the NBR feature to the intersection of the NBR of the
            Determiner, Nominal Head, and NCOMP;
Otherwise,
            set the NBR feature to the intersection of the NBR of the
            Determiner and Nominal Head;
If
            The Nominal Head is singular and not a mass noun
Then
            Label the NP with the FACTOR UNLIKELY, because no determiner
(1980:60)
```

Because this approach is rule based, it is up to the rule programmer to notice ill-formed inputs, develop rules to parse them, and score them. For example, the same rule system rejects NPs such as *two boys than that* or *many water*. This is typical of a construction-oriented method. There are no general principles that explain why a particular construction fails to pass all grammatical constraints, aside from the descriptivist statements like mass/count noun disagreement.

The second approach is typified by the work of Weischedel and Black (1980) and Kwasny and Sondheimer (1979). Weischedel and Black were working within a foreign language instructional system, in which ill-formed inputs made by beginning students were to be expected; for instance, they cite the example of English-speaking students forgetting "to put past participles at the end of a clause" in German (1980:99), thus using *Ich habe gegessen das Fleisch* rather than *Ich habe das Fleisch gegessen*. They offer, first of all, to simply add more rules (the first approach described above): "if particular forms are anticipated, they may be explicitly included in the syntactic model ... the path in the augmented transition net (ATN) corresponding to the incorrect form computes a message to tell students of the mistake." (1980:99).

They also propose that one simply relax the predicates attached to network transitions:

> A straightforward example of the use of such predicates is for subject-verb agreement. It is easy for a user to make mistakes in long English sentences, resulting in parser failure. A solution would be simply to remove the predicate from the rule. However, Grishman (1973) reports from their experience ... that the predicates effectively eliminate a large number of spurious parses.
>
> We suggest that, instead of forcing all predicates to be satisfied or ignoring

> the information inherent in them, that the designer should designate that *certain predicates can be relaxed*, with a record being kept of each predicate not satisfied during parsing. Only parses yielding the fewest unsatisfied predicates are completed. Since the number of predicates that evaluate to false in a partial parse is a nondecreasing number, only those partial parses with the fewest unsatisfied predicates have to be continued ...
>
> The notion of allowing certain predicates to go unsatisfied is much more general than the highly special environment of the German tutor. ... several predicates were made optional or "failable." By "failable" we mean that the predicates ought to be true for the pattern to match, but could be false without preventing the pattern from matching if there would be no parse at all with such predicates true. In addition to subject-verb agreement, pronominal case was also made failable. The two together allow a sentence such as "me think him win often" to be parsed, even though the parser has a model of correct language. (1980:99-100).

Note that this model comes closer to the principle-based ideal: in effect, a surface construction is allowed if it passes all well-formedness constraints; thus, by relaxing constraints one by one, we get a graded set of surface sentence possibilities. What is missing is a principled way to know what constraints to relax. In addition, models of this sort generally limit themselves to the obvious predicates placed on ATN arcs—agreement and cooccurrence restrictions—and not the full range of constraint violations (like island violations or thematic violations). For example, Kwasny and Sondheimer's (1979) model for handling ill-formed input similarly works by "selectively relaxing predicates to deal with co-occurrence violations and relaxation of expected word categories." In short, these construction-oriented approaches to not give a *theory* for ill-formed input; nor would we expect them to.

A principle-based approach confronts the problem of ill-formed input head on, and says directly what constraints are violated by any particular sentence. As a concrete example, suppose we take the following (familiar) modules as central to sentence well-formedness at the syntactic level (ignoring for the moment phonological and logical form). This list is not meant to be exhaustive, but simply to indicate the range of government-binding theory modules. Nothing hinges on their details here.

1. Theta principle: Essentially verb subcategorization. All arguments must bear one and only one thematic role (such as Agent, Patient, Theme), assigned typically by a verb. Thematic assignment is carried out under government (see below).
2. Trace theory: Traces must be properly governed; they do not get case marked.

3. Bounding theory: No trace can be more than one bounding node from its antecedent. (More recent version: every link of a chain must be subjacent to the next link.)
4. $\overline{\mathrm{X}}$ theory: all phrases are projections of $\pm$N, $\pm$V, under the usual strict interpretation, we may say that only maximal projections are subject to movement.
5. Projection Principle: all levels of representation obey the θ principle.
6. Empty category principle: A trace must be properly governed.
7. Government: the basic structural relationship between phrases; X governs Y iff X c-commands Y and Y c-commands X (this is just one variant definition).
8. Case Theory: overt arguments must be case marked; or if an argument is not in a position to which Case is assigned, the resulting structure is grammatical.
9. Binding Theory: a moved *wh* element c-commands its gap; various other (co-indexing) relationships between pronouns and their antecedents.

We can now classify sentences as to whether they pass all or some of these constraints. Parsing so-called ill-formed sentences amounts to applying the well-formedness conditions above; the more constraints a sentence violates, the more ill-formed it is. Of course, matters are more complicated than this; for example, simple agreement violations seems to cause more problems than they ought to. Given this, Crain and Fodor (1985) argued that the real problem has to do with confusability: subjects are looking for a near-miss grammatical example, and that's possible with agreement violations and not possible with violations such as *Who did you see Picasso's picture of*. Therefore, in the agreement violation case, processing will be longer as the subject searches for a possible alternative; in the second, constraint violating example, the subject gives up rapidly. Freedman and Forster (1985), in their published article, attempt to lay this matter to rest by trying to show that correctability does not account for all such results, but from my point of view the entire matter still seems unclear.

In any case, we do know that certain constraint violations don't disrupt processing fluency and others do. My own hunch is that the answer will again hinge how the information modules are temporally wired together: (1) $\overline{\mathrm{X}}$ processing does occur rapidly and autonomously, and violations of such constraints force major disruptions. (2) Agreement violations cause processing hiccups because of the correctability effect, but this processing is still done modularly (as nearly everybody in computational linguistics wants it done);[47]

[47]Since this is not an exhaustive list, certain "ungrammatical" sentences, in particular, agreement vi-

(3) Case and thematic violations are next worst; (4) Locality and Binding constraint violations cause the least processing disruptions because processing has already passed to a second stage of interpretation by that point; (5) However matters work out, Freedman and Forster (1985) discovered that nongenerable strings are processed less fluently than strings that pass some, but not all, constraints, and we should now abandon the whole notion of generation and replace it with constraint satisfaction.

As one might expect, since some principles (like binding) depend upon calculations that other licensing principles provide, one might expect that some constraint violations are less harmful than others. For example, an $\overline{\text{X}}$ violation may prevent one from even being able to compute structural relationships among the elements in a sentence, and so tends to result in a completely uninterpretable sentence. Similarly, thematic theory violations can't be as easily interpreted since missing arguments can't be supplied. On the other hand, agreement, binding or bounding violations must be calculated relative to an already licensed structure, and so are usually interpretable. For example, *John seems it is certain to like ice-cream* or *Bill thought Mary likes himself* are both parsable and interpretable. In this way, a principle-based account provides a *theory* of ill-formed sentences in a way that a rule-based approach does not.

1. Violation of theta theory: *John hit*; *Bill hit John Joe.*
2. Violation of trace theory: *John hit t*
3. Violation of bounding theory: *The man$_i$ who pictures of e_i are on the table*
4. Violation of $\overline{\text{X}}$ theory: (More than one head in a phrase) *John Mary kissed Sue.*
5. Violation of Projection Principle: *John hit e_i.*
6. Violation of Empty Category Principle: *Who$_i$ do you think that left e_i.*
7. Violation of case filter: *John's destruction the city*; *I bought quickly the book* (Adjacency violation in English).
8. Violation of binding theory: *John said that Mary thought that washing himself would be OK.*

olations that are not accommodated by any of the modules listed above, are not captured. Presumably, simple Subject-Verb agreement could be encoded in the usual way as (atomic) feature unification between INFL and Subject in English. Crain and Fodor's objection to such results argues that agreement violations Such properties might or might not be base generated, as Freedman and Forster note.

Freedman and Forster (1985) also want to find out where such violations are processed—at S-structure, LF, D-structure, or somewhere else. Their tentative answer pins it down to a blend of S-structure and "plausibility." This is such a rough-cut answer that it would fit almost any model. For example, one could take S-structure to be just that provided by a completely multiplied-out set of GPSG rules, and "plausibility" to be whatever it is. Clearly, much more detail work must be done here: can one use quantifier effects to dissect out different levels in LF, for example, as Hornstein (1986) proposes? A glance at table 1, which gives a more complete listing of GB principles parameters (under one account of GB theory!) shows that a wide range of hypotheses could be entertained.

How would a parser accommodate these constraint violations? Freedman and Forster (1985) observer that a rule-based parser, for them, the Marcus parser, would be hard pressed to maintain an account of sentence overgeneration. But if we follow Kashket's model described in the preceding section, then nothing special need be said at all, though the parser may fail at various points because it winds up with incomplete subtrees at the lexical level or with an unattached subtree at the phrasal level. For example, if Kashket's parser is given a sentence with a verb that must assign DATIVE case and there is only one NP case marked ABSOLUTIVE, then case assignment fails and the NP remains detached. Thus this NP is not licensed, even though the sentence is partially analyzed. Exactly as much of the structure as can be built is built.

The nonhomogeneity of principle-based systems assists here as well. If we assume that *wh-movement* well-formedness is checked by a separate parser component (as in fact most systems do: consider ATNs, the Marcus parser, and Abney's licensing model described in the next section), then we would expect there to be sentences that can be completely analyzed phrasally, but cannot be assigned correct *wh*-empty category indexings, such as *What do you wonder who likes*—roughly because all the parsing machinery aside from the separate *wh* component works fluently.

This is not to say that a rule-based system cannot reproduce the effect of graded unacceptability. After all, the ATN and Marcus systems are "rule-based" in the traditional sense of the term. But I would argue that to the extent that they factor out a distinct operating unit with its own behavior that is imposed on top of the basic parser operation—be it hold cell or wh-comp mechanism—these parsers slide at least part of the way into the principle-based domain. Note further that this is also true of agreement violations: since agreement is typically imposed nonhomogeneously on top of otherwise well-formed tree structures, agreement violations may be understood as principled violations of well-formedness conditions, precisely because they can be factored apart from phrasal well-formedness. Indeed, to the extent that we break apart a surface construction into the interaction of several components, we adopt a principle-based view.

$\bar{X}$ Principles	*$\bar{X}$ Parameters*	*Level of Application*
A phrasal projection ($\bar{X}$) has a head (X), a specifier and a complement	Constituent Order, Basic Categories, Pre-terminals, and Specifiers	DS
Left and right adjunction are optional and occur on the X^{max} or X^0 level	Choice of Adjuncts and their positions	DS
Complements of nonlexical heads (*e.g.*, C and I) are determined from default values	default value for nonlexical heads	DS
Specifiers may be optional	optional specifiers	DS
Complements of lexical heads are projected from argument structure information	Subcategorization Information (Lexicon)	DS

Government Principles	*Government Parameters*	*Level of Application*
α governs β if α is a governor and α minimally c-commands β	Choice of governors	SS, DS, LF

θ Principles	*θ Parameters*	*Level of Application*
[CL $+case_i$ $+\theta_j$] ... [NP $+case_i$] $\Rightarrow$ [CL $+case_i$ $+\theta_j$] ... [NP $+case_i$ $+\theta_j$] if language allows clitic doubling	Clitics, Clitic Doubling	SS, DS, LF
θ-Criterion	no parameter	SS, DS, LF
Visibility Condition	no parameter	SS, DS, LF
CSR(*semantic role*) = *syntactic category*	CSR Mapping	SS, DS, LF

Case Principles	*Case Parameters*	*Level of Application*
(A) Objective Case is assigned to object governed by transitive P	Choice of Government	SS
(B) Objective Case is assigned to object governed by transitive V	Choice of Government	SS
(C) Nominative Case is assigned to subject governed by [Infl +*tns*]	Choice of Government	SS
Case Filter (obviated by Visibility Condition)	no parameter	SS

Trace Principles	*Trace Parameters*	*Level of Application*
V, A, N, P, N_i and NP_i are proper governors for ECP (where EC = $[e]_i$)[1]	Pro-drop	SS
An empty category $[e]_i$ with [−V] feature must be properly governed (PROP is ungoverned)	Choice of Traces and ECP Chain Conditions	SS (and possibly LF)

Binding Principles	*Binding Parameters*	*Level of Application*
(A) An anaphor must be bound in its governing governing category	Choice of Governing Category	LF (if lexical) SS (if trace)
(B) A pronoun must be free in its governing category	Choice of Governing Category	LF
(C) An r-expression must be A-free (in the domain of its operator)	no parameter	LF (if lexical) SS (if trace)
V, A, N, P and [Infl +*tns*] are governors for Binding	no parameter	SS, LF
A variable must be A-bound	no parameter	SS

Bounding Principles	*Bounding Parameters*	*Level of Application*
Move-α may cross at most one bounding node	Choice of Bounding Nodes	DS $\rightarrow$ SS

Table 1: Principles and parameters in one version of GB theory.

Once again then, the difference between rules and principles here is not so much one of formal distinction as it is practice: how *easy* is it to accommodate ill-formed sentences in a given framework? If we have not factored apart the distinct constraints that contributing to an unacceptable sentence, then it may be particularly hard to unravel their effects, and this is why a principle-based parser gains in handling such cases. By way of example, consider what one must do with a traditional context-free grammar to parse a sentence such as *Mary to like ice-cream is nice.* One must add a new rule that allows *for* to appear optionally in a sentential NP, along with some flag so that we know that such a sentence is not perfectly well-formed:

Sentential NP$\rightarrow \bar{S}$
$\bar{S} \rightarrow$ for NP INFL-tns VP
INFL-tns$\rightarrow$ to
VP$\rightarrow$ like NP
$\bar{S} \rightarrow$ NP INFL VP
  new rule (score 0.7)

The problem is that the new rule really doesn't say *why* the sentence is bad—it's just mixed in with the other rules. It's another example of having the wrong vocabulary to talk about cause and effect: what we'd really like to say is something to the effect that *Mary* needs to be case marked by *for* (given the absence of a tensed verb). On this principled account, we don't have to add a new rule at all. Furthermore, consider what happens when we try to extend things to encompass a closely related example like *Mary likes ice cream is nice.* On the context-free rule approach, we have to add another new rule that would permit the absence of *that.* This rule would be completely distinct from the rule to handle an optional *for*—yet we know that the two problems are closely allied, since the same constraint that is being violated when we omit *for* is at work when we omit *that.* A full context-free system to handle all possible unacceptable sentences would be enormous. Note that it could not simply say that all sentences are acceptable—this would take a very small grammar, of course—because we would still like to recover the proper structural descriptions of completely well-formed sentences, and judge gradations in unacceptability. The scoring functions would have to be quite carefully hand-tailored to achieve this goal. In contrast, a parser using case-marking principles would be able to report back immediately where and why such sentences are unacceptable, with no additional machinery.

Principle-based handling of unacceptable sentences is not confined to the GB model, of course. The same remarks apply to any theory—like GPSG—that possesses well-formedness constraints. Figure 14, taken from Sells (1985:79) outlines the major GPSG components. We see that while GPSG does contain rules (although they are in fact admissibility constraints on well-formed subtrees, hence closely allied to constraints), GPSG also contains a variety of principles that apply to an expanded set of immediate dominance rules and relate these to well-formed syntactic structures: feature cooccurrence restrictions; feature specification defaults; the Head Feature Convention; the Foot Feature Principle; the Control Agreement Principle; and linear precedence statements. By factoring apart the well-formedness of individual constructions into several components, GPSG can also isolate unacceptability into one or another component, and produce a parser that can automatically handle ill-formed sentences by relaxing one or another of the constraints. The information flow inherent in the GPSG picture also leads to a natural (though perhaps incorrect) picture like the GB one: certain constraints lead to gross unacceptabilities, because they do not even license correct phrasal trees; while other constraints violate only agreement conventions and so permit phrasal construction to proceed unhindered all the way through to the bottom of the GPSG information flow diagram. I see nothing then that bars GPSG from developing a principle-based treatment of unacceptable sentences.[48]

In short, when we have a principle-based parser, we need only state a constraint once, and we know why an unacceptable sentence is unacceptable. With a rule-based system, we in effect work with the theorem-like results of deductions from more basic axioms, and so we lose the intermediate steps that tell us why such sentences are unacceptable (other than the scoring function which is left behind). Given a construction-oriented approach, there is a further temptation to simply dump in another rule to cover each unacceptable case that comes along, without recognizing the systematic patterns that lie behind the unacceptabilities. As is well known, simply adding more rules to a context-free grammar can have unintended consequences (ask anyone who has written a large context-free grammar.) Finally, readers familiar with AI expert systems should recognize immediately the superiority of a system that can reason from first principles in order to understand what has gone wrong.

4.2 Principles and the role of core vs. periphery in grammar

How might we expect the distinction between core and peripheral constructions to be reflected in a principle-based parser? Should stylistically marked or idiosyncratic con-

[48]Of course, there may well be more subtle cases where the distinct vocabularies used by the two theories leads to a different taxonomy of unacceptable sentences. This may well be true in the case of long-distance dependencies, if they are handled by passing SLASH features through phrasal trees in GPSG. I will simply take note of such possible differences here, pointing out that the basic strength of the principle-based approach stands no matter which of these theories one adopts.

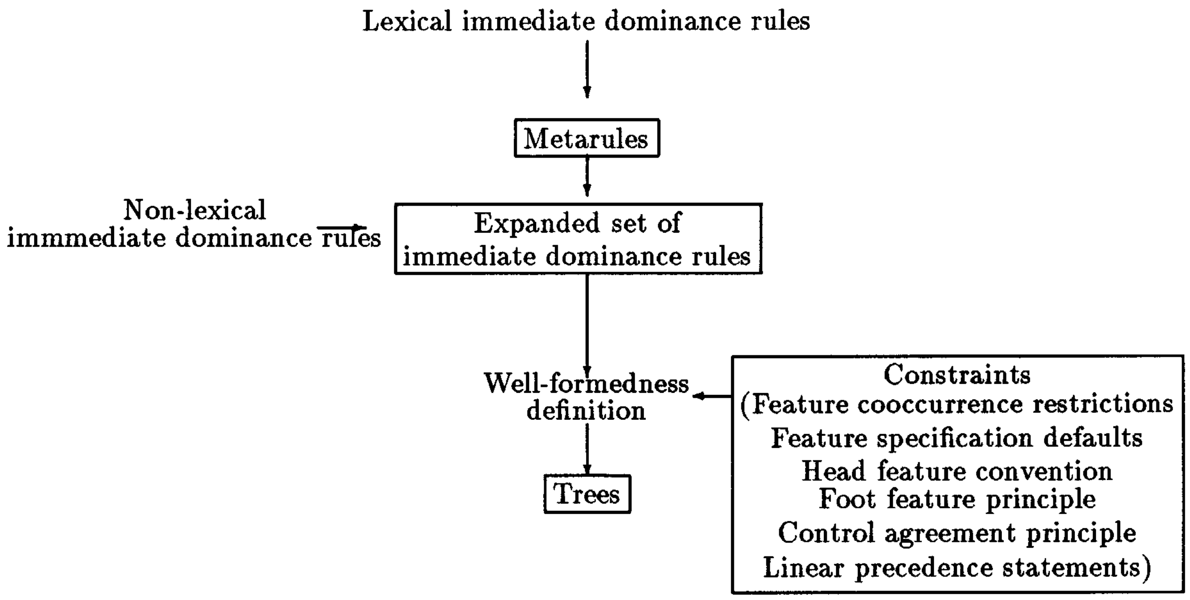

Lexical immediate dominance rules
Metarules
Non-lexical immmediate dominance rules
Expanded set of immediate dominance rules
Well-formedness definition
Trees
Constraints
(Feature cooccurrence restrictions
Feature specification defaults
Head feature convention
Foot feature principle
Control agreement principle
Linear precedence statements)

Figure 14: GPSG builds up surface constructions also by superimposing a set of constraints.

structions be more or less difficult to process? Since the basic information flow of the principle-based model leads us to first establish licensing conditions, given a principle-based parser like Kashket's (or the Abney or underspecified $\overline{\text{X}}$ model described in the next section), we would expect sentences that meet basic licensing conditions to be relatively easy to process, while more "marked" constructions would be those that violated basic licensing conditions (government and the like). The implications for processing complexity are unclear. For example, infinitival sentences with lexical subjects (*I want John to win*) meet all the basic licensing constraints, and I would not expect them to cause any extra processing difficulty. This prediction contrasts with that offered by Frazier (1986), who implicitly argues that a direct, principle-based parsing system would predict an extra processing load in such cases.

I see no reason why this extra processing load should necessarily follow. While it's true that examples such as *I want John to win* are more unlikely, that doesn't mean that they're hard to parse via direct $\overline{\text{X}}$ projection. Section 5 below shows that simply by projection, in turn, maximal phrases for *John*, the inflectional element *to*, and the verb *win* will do the trick: the expectation that *want* takes a case recipient will permit it to actually assign case to either *John* or the full sentence, depending on which sequence of words actually appears. Such an example should be just as hard to parse directly as *John wants ice-cream*—which is what Frazier claims to be the case (1986:24).

In fact, Frazier would like to say more generally that there is no processing distinction between core and peripheral constructions (1986:24)—and that this provides an argument for a "compiled" rather than a "direct" use of grammatical principles:

> ... we might expect infinitives with lexical subjects to be difficult to understand. Presumably derived nominals with "of" inserted (e.g., destruction of Rome) should also be perceptually complex, assuming "of"-insertion is idiosyncratic and thus peripheral (i.e., we can't simply list it and let it apply to any caseless noun phrase such as "John" in "It was hit of John" or "I tried of John to win.") Similarly phrases with idiosyncratic order ("someone clever" cf. Emonds 1986) should be difficult to parse, on the view being considered.
>
> There is evidence suggesting many so-called 'stylistically-marked' constructions are relatively difficult to parse, at least in isolation. But in every case I'm aware of, the complexity of such constructions can either be attributed to independent parsing principles and/or its complexity is eliminated when the sentence is presented in the appropriate discourse contexts. (1986:41)

The problem with Frazier's point here, I believe, is that her examples of 'marked' constructions really aren't marked with respect to a principle-based parser. We have already covered the example of lexical subject infinitivals. For *of*-insertion, it also seems likely that licensing processing will proceed normally: the parser will recognize a nominal form, as in *destruction*, via projection from the lexicon; the next token, *of*, assigns case normally to the NP to the right, and no unusual processing at all takes place. On the other hand, *It was hit of John* will cause the processor to balk (as expected and as seems true on introspection), since *was hit* will not be processed as a nominal (+N) form; it does not seem outlandish that this idiosyncrasy would be encoded in lexicon and accessed as such directly by the parser, demanding a caseless NP to the right.

5 Parsing models for principle-based systems

We turn next to parsing models for principle-based grammars, contrasting them with rule-based systems. We have already seen how Kashket's parser works by imposing well-formedness conditions on words as they enter a hypothetical input stream. The chief difference from a rule-based parser, as one might expect, is that a principle-based parser proceeds by checking a variety of heterogeneous well-formedness conditions as it goes along, rather than "matching" against a storehouse of uniform individual rules. (A context-free rule system represents a paradigm example of such a matching system.)

But there are other issues to address. Since a principle-based need not recover phrase structure, this leaves open the question of exactly what kind of information structure must be built. Here there are many open possibilities; this section will consider just a few. First, as in Abney's system, one might simply recover licensing relations directly. Second, as in Kashket's parser, one could just recover adjacency relations at one autonomous level and hierarchical relationships at another. Third, one could assume a predicate such as *government* to be basic to the parser's job. On this model, government relations could be built via skeletal (topological) projections from $\overline{\text{X}}$ theory, supplemented by intonational phrasings (as supplied by the speech stream). We'll take a brief look at this in section 5.2 below. Fourth, one could assume a basic pushdown stack model that extensionally provides the equivalent of c-command, as suggested in Berwick and Weinberg (1984). If we take government to be mutual c-command, then such a model would already have at its disposal much of the apparatus to compute the well-formedness conditions for syntax. (In a sense this is what Abney's parser does.) Fifth, one can interleave $\overline{\text{X}}$ projection along with application of other well-formedness constraints; this is the approach that Dorr takes for translation parsing, described below in section 6.

A second set of issues centers on how we carve up the information structures needed for parsing. On the most transparent view, we simply "use" grammatical principles directly. Section 2 covered these points, but we'll review them here.

For example, in the case of $\overline{\text{X}}$ theory, this would mean projecting lexical Heads as they are encountered in the input: using old-style terminology, we would build a VP as soon as a verb is found, and an NP when an N is encountered, and so forth.[49]

But this is a static view. If we distinguish the abstract computational problem from its algorithmic solution, then other direct solutions are possible. For example, in the case of $\overline{\text{X}}$ theory, one might reasonably split up a direct implementation into two steps: (1) projection of basic $\overline{\text{X}}$ templates, and (2) filling in the features of heads.

[49]Abney actually takes Det as the Head of NPs and so would behave somewhat differently.

Computationally, there's no reason why these must be taken together, and, in fact, Dorr's parser separates them. Yet such an approach still faithfully and transparently reflects $\overline{\mathrm{X}}$ principles. In short, while the logical structures of $\overline{\mathrm{X}}$ theory are still respected, the temporal flow of information is split into two parts: topological projection and feature identification. Since the grammatical theory almost by definition says nothing at all about the temporal organization of parsing—it simply defines an abstract computational problem, in Marr's sense—we would almost expect a principle-based parser to have some freedom in the way that it temporally exploits the information structures and principles of that theory, as long as it respects the theory's information structures. There's no *necessity* to combine strict projection from lexical items together with $\overline{\mathrm{X}}$ topology, as argued, for example, by Frazier (1986).

If one is willing to abandon traditional phrase structure, then even more radical changes are possible. As suggested by Frazier (1986), one could take the notion of *chain* as the single basic structure recovered for syntactic analysis—the parser must build up chains of grammatical elements, and all other grammatical relations have to do with either the heads of chains (as in logical form, the operator positions); the tails of chains (d-structure, or thematic positions); and tails of chains and lexical elements (s-structure).

In brief, the point being made here is not whether one or another of these positions is correct, but rather than our rule-based blinders have led us to neglect a world of possibilities. The door to information-structure alternatives is flung wide open by considering principle-based parsers.

Perhaps, though, the door has been flung open too fast: won't the uniform matching we're so comfortable with in context-free parsing always be faster, precisely because it is uniform? And won't principle-based parsing always be slower precisely because it must laboriously deduce from an axiom set whether a surface construction is possible?

One half of this question is answered by looking at specific examples. In this section we will see via explicit case studies that direct principle parsing need not slow things down, contrary to some expectations. Indeed, principle-based parsing appears to resolve certain problems that plague deterministic parsing. In addition, they open up a wider range of parsing techniques than variations on context-free parsing methods like the Earley algorithm, CKY parsing, or its Prolog-based analogs. In part, this is because we may eschew traditional tree-based predicates in favor of whatever predicates are needed to compute well-formedness conditions—such as c-command and government. We may in fact take the hallmark of a principle-based parser design to be its avoidance of traditional phrase structure, since this is largely subsumed by other predicates in such a system.

Finally, in the conclusion of this paper we will turn briefly to the second part of an answer to the processing speed question: are principles "compiled" into rule-like form? We will see that an autonomous processing theory for principles—grounded on Halle's spiral notebook conception for linguistic structure—actually suggests more efficient parallel processing schemes for modular, principle-based parsers. The information structure of principles does suggest, however, that there are deductive clusters that may *look* like organized, compiled, rule systems.

We will only cover two basic ways to organize a principle-based parser here, focusing in detail on two specific proposals. There are other possibilities that we will not describe in detail. Sharp (1985) and Thiersch (1986) have built GB parsers using Prolog to encode the required well-formedness predicates (axioms), while relying on Prolog as a backtracking proof procedure to deduce whether a given sentence can follow from the axioms.[50] While these are perfectly acceptable approaches, they do have Prolog carry out the bulk of the computational work, and so obscure some of the computational issues that we would like to explicitly address here. (It is interesting to note that Thiersch's system, for example, does not seem to suffer from any lack of speed; in fact, subject to the usual caveats about unequal coverage, in real-time comparisons against a large context-free German grammar implemented at MIT, both running on an AT-type machine, it does quite well. Evidently Prolog is fast enough so as not to lose anything by forcing one to actually deduce surface constructions from more basic principles.) Wherli (198x) has built a Pascal implementation of a GB parser. And there must be others lurking about in the depths of middle Europe.

Because we are interested in the computational issues that these Prolog designs sidestep, we will not cover these two principle-based systems here. Instead, we analyze two other alternatives that grapple more directly with the issue of principle-based processing: Abney's licensing model and Berwick and Barton's $\overline{\text{X}}$ parsing model.

5.1 Parsing as licensing: Abney's model

In recent work, Abney (1986) has developed a GB-based parser quite similar to Kashket's in the sense that it is grounded on the notion of *licensing*: every element in a sentence appears there because it is fulfilling some grammatical role—entering into a grammatical relationship either on the giving or receiving end—and a sentence is well-formed if and only if every one of its elements is *licensed*. Also, just as in Kashket's parser, case relationships are assigned essentially under sisterhood, without necessarily having to first build $\overline{\text{X}}$ skeletons. We saw how case assignment licensed elements in Kashket's Warlpiri parser. In fact, Abney's parser never really builds phrase structure at all. Figure 1 showed

[50]It is a remarkable irony that formal language theory was invented over thirty years ago using this parsing-as-deduction method.

how Abney's licensing relations of thematic assignment (θ), subjecthood (S), functional selection (F), and modification (M) work to validate every element of a simple sentence.

In particular, for Abney licensing relations are: (1) unique; (2) local (in effect, operating under sisterhood, just as Kashket claims); and (3) lexical (particular verbs assign particular thematic roles). Importantly, Abney's model shows how one can apply case and thematic principles *before* phrasal structure is completely fixed; indeed, in his system, it is case and thematic licensing that fix phrasal structure. Additionally, his parser includes a blend of heuristic-like principles (it analyzes roughly two words at a time) that are presumably derived from the competence grammar itself. Thus, Abney's parser is not a "direct" principle-based parser. Nonetheless, it is clearly principle-based.

5.1.1 Abney's parser in action

We can now review in more detail the operation of his parser on the sentence *John will like pictures of Mary*. Like Kashket's parser, Abney's works on basically two words at a time, without referring to any explicit phrase structure rules. Abney assumes the following assignment of licensing relations to lexical items. (S, F, and M stand for the three varieties of licensing besides theta assignment; when theta assignment is mentioned, the particular thematic role involved is listed. The directionality of assignment—since Abney is talking about English—is also listed.)

will:	$\langle\leftarrow$ NP S $\rangle$
(*will* assigns Subjecthood relation to NP to its left)	
will:	$\langle\rightarrow$ V F $\rangle$
(*will* functionally licenses a V to its right)	
like:	$\langle\rightarrow$ NP THEME $\rangle$
(*like* thematically licenses an NP theme to its right)	
pictures:	$\langle\rightarrow$ [genitive] THEME $\rangle$
(*pictures* thematically licenses a genitive theme to its right)	
of:	$\langle\rightarrow$ NP F $\rangle$
(*of* functionally licenses (case marks) an NP to its right)	

As Abney then points out,

> We can parse our sample sentence by means of a very simple algorithm: proceeding from left to right, examining two sentences at a time, assemble them by assigning whatever relations can be assigned. The parse would proceed as

```
Step 1:    John        will
            [N]         [I]nflection
                       < <-- N  S  >
                       < --> V  F  >

Step 2:   John<---license----will         like
          [N]     <---N  S    [I]           [V]
                            < --> V  F  >  < -->  N  theta  >


Step 3:  John<---will----license--->like      pictures
                 [I]                [V]          [N]
                               <--> N theta>    <--> [gen] theta  >

Step 4: John<---will--->like---license--->pictures    of       Mary
                 [I]     [V]                  [N]       [P]       [N]
                                                 <--> NP  F >

Step 5: John<---will--->like--->pictures    of---license--->Mary
         [N]     [I]     [V]      [N]       [P]               [N]
```

Figure 15: Parsing by 2-word licensing.

> follows: ... *John* has no roles. Only the first of *will*'s roles is assignable to *John*; the second matches in neither directionality nor description of receiver. ... At each point, exactly one role is assignable, making for a rather trivial parse. (1986:5)

Figure 15 shows the snapshot sequence for the parse. What is more, as Abney states,

> Once the licensing structure of the sentence has been recovered, phrase structure follows straightforwardly. ... It is also reasonable to suppose that licensing relations, taken as a group, are nonreflexive and noncyclic: i.e., no element can be licensed by itself, and no element can be licensed by another which it itself licenses (directly or indirectly). This guarantees that licensing relations, taken together, define a finite tree. If we assume that all relations are associated lexically with their assigners, then only words can be assigners of licensing relations ... Finally, assume the relevant locality condition on licensing relations to be strict government: mutual c-command. If we eliminate, or at least ignore, inter-

> mediate bar level categories, this locality condition is simply phrase-structural sisterhood. (1986:5–6)

If licensing relations are so very much like phrase structure rules—akin to the McCawley type well-formedness templates for depth-one trees—then the right question to ask is whether this principle-based approach is any different. The answer is yes: licensing relations are stricter than phrase structure templates because certain $\overline{\text{X}}$ constraints are enforced—there is a unique head and "all nonheads are maximal categories." Abney observes that licensing, as expected, also works better across languages: while there need be no connection between the phrase structure rule from one language to another, this is not true for licensing. Because the licensing vocabulary is so restricted, the only thing that can change is basically the direction of licensing (right or left). This greatly narrows the possible class of licensing relations (hence phrase structure realizations).

It is also interesting to compare this system to a much older framework: categorial grammar. In a certain sense, categorial grammar does the same work, telling us what can combine with what. As expected, a key difference is that Abney's system is grounded on principles: we do not have arbitrary combinatory rules, but we do have a more restricted vocabulary of licensing direction and licensing relationships. This eliminates the need for any kind of complex cancellation "arithmetic." (On this view, categorial grammar superficially reflects these more basic combinatory principles.)

As one can see, Abney's parser assembles subtrees in a roughly bottom-up fashion. No rules are used, however, but only general admissibility constraints. To deal with a wider range of sentences, Abney's full parser includes a much richer array of processing techniques, which we review only briefly here.

One of these deals with top-down expectations. For these, Abney adds *templates*, basically, partial government fragments. The problem arises because the parser must sometimes wait until a licenser appears: for instance, an adjective following a determiner must wait until it sees its noun licenser. To handle such cases, one can add to the determiner a template [Adj Noun]—the expectation that the adjective will indeed be (ultimately) followed by a noun. This template is assigned by the determiner to the adjective. (Presumably this would also hold for multiple adjectives.) Abney observes,

> Templates have an obvious similarity to phrase structure rules. In fact, they perform precisely the function which phrase structure rules perform. The important difference between this system and systems based on phrase structure rules is that in such systems, the phrase structure rule format is used as a homogeneous rule format for all aspects of parsing. Here, templates are only one segment of a larger system, and are highly specific to a certain task, rather

than being used for tasks for which they are not suited. (1986:14)

Frazier (1986:17) argues that Abney's use of templates really means that one isn't using grammatical principles directly at all any more. But surely this is a matter of degree. Templates are quite limited in their activity—they apply in only very special, isolated circumstances. In this respect, they may indeed reflect heuristic parsing residues.

A second set of additions center on structural revisions necessitated by incorrect parses—Abney's parser backtracks to rectify incorrectly assembled licensed structures. This will happen, for example, in well-known sentences where a verb can subcategorize for either an NP or a sentence, such as *They knew the solution to the problem would be easy.* Abney hypothesizes that such seemingly fluently processed cases are corrected by purely adjacent, local reorganizations of the licensing structures. I think this view to be largely correct, and in section 5.2 we will take another look at such examples.

Third, Abney uses his licensing principles to resolve some notorious cases of attachment ambiguity. His approach here does not bear directly on the issue of principle-based parsing, except to point out that the licensing vocabulary may provide a better account of such cases than previous, purely structural (and essentially context-free rule based) approaches. If more than one node in the preceding near edge can license the current node (as in *I ate the ice-cream on the table*), then Abney's parser chooses one based on the following ranking:

1. θ-licensers preferred over non-θ-licensers;
2. verbs preferred over other categories;
3. low attachment preferred.

This ranking is partially principled because θ-licensers and verbs play a central role in validating a fully-licensed structure: without a verb, or θ-licenser, arguments are not assigned case. Finally, the third action may be regarded as a default that is taken when no licensing priorities apply. Thus, unlike systems that aim at purely descriptive statements of attachment preferences (such as minimal attachment or processing strategies such as ordering PUSH vs. SCAN operations in an ATN), this ranking is highly rationalized. Again, the point is not that processing strategies can't simulate these effects; the point is that principles guide us to the appropriate solution. And Abney's ranking fares reasonably well: as he points out, it correctly predicts the preferences in examples such as *I could interest the mayor in a new Volvo* (θ licensing wins over modification); *I thought about his interest in the Volvo* (θ marker *interest* wins out); *I sang to the cat in the kitchen* or *I* *wrote a letter to a friend* (both candidates θ mark but the verb wins out); and *a gift to a boy in a box* (both θ licensers are nonverbs, so default low attachment wins, even though the fragment is semantically anomalous interpreted this way).

Finally, Abney's parser does what all principle-based systems ought to do: it contains several different devices to follow the "layer-cake" view of sentence constructions, rather than attempt to shoehorn everything into a single kind of rule format. In particular, like many previous parsers, Abney's contains a distinct device for handling displaced constituents. However, he has turned this into a separate movement module, that is not concerned with licensing or structure building at all, aside from inserting empty categories. In this Abney follows the Berwick and Weinberg two-stage approach to parsing: the first stage builds correctly licensed structure (phrase structure in Berwick and Weinberg's sense, though this is replaced here with licensing structure); the second stage uses that structure as a scaffolding on which to assign filler-gap relations.

The constraint that an antecedent c-command its gap is quite transparently encoded via the licensing relations. Abney does not take this view, but, recalling that our goal is to reduce all grammatical relationships to adjacency, consider a sentence such as *Who did you think Bill saw*. Abney's model will license the following grammatical relations, where IP stands for an inflection projection and CP for a complementizer projection:

[Who→IP-did]	[NP-You←Infl-did]	[Infl-did-→think-V]
[think-V-→CP]	[C-e-→Infl-saw]	[NP-Bill←Infl-saw]
[V-saw-→e]		

We note that there is an unbroken chain of *adjacent* licensed grammatical relations extending from *who* down to the gap after *saw*. Put another way, the composed, adjacent licensed relations are the reflex of what has been called a path or a chain in the theory of filler-gap relations. The licensing principles transparently enforce the c-command condition.

When this chain of adjacencies is broken, this corresponds to an (ill-formed) sentence where an antecedent does not c-command its gap, as in Abney's example *Who*$_i$ *Bill saw surprised* e_i. Here, there is no grammatical relation linking *who* with the chain leading from CP through the verb *surprised* down to a gap.

[Who-→IP-saw]	[NP-Bill←Infl-saw]	[V-saw]
[CP←Infl-surprised]	[V-surprised-→e]	

If this approach is correct, once again no additional rules need to be added in order to represent the correct licensing structure for long distance dependencies; only the basic principles themselves are needed. We may contrast this approach with one that attempts to multiply out into the rule system itself the required chains. As before, if one attempts to do this using *one* kind of rule, then one first of all blows up the number of rules one needs enormously (see Barton, Berwick, and Ristad 1987 appendix B for an estimate in English); second of all, one folds together two distinct phenomena: local licensing and filler-gap dependencies. This is no different from trying to encode agreement relations in context-free rules: the result is a multiplied-out rule system that tries to do too much with a single rule format. In principle-based parsing, the ideal is to operate a number of distinct modules, each with its own particular informational constraints and independence.

5.2 Direct parsing of $\overline{\text{X}}$ rules

As our third example of principle-based parsing, we will consider what it means to parse using $\overline{\text{X}}$ theory directly. Our goal will be a familiar one: to show that if one embeds $\overline{\text{X}}$ theory *directly* into a parser instead of using multiplied out context-free rules, one in fact gets a better deterministic parser. The model here is based on Berwick and Barton (1985) and Berwick and Weinberg (1985).

> The direct use of $\overline{\text{X}}$ theory permits one to delay parsing decisions so as to retain the advantages of deterministic parsing without any special machinery of the kind proposed by Marcus (1980). In particular, no special "attention shifting" mechanism is required.
>
> The parser does not use any lookahead, yet remains faithful to human processing of "garden path" sentences. It fails naturally on "short" garden path sentences (such as *The boat floated sank*) that were perfectly well parsable using the Marcus 3-cell lookahead design. At the same time, the $\overline{\text{X}}$ format succeeds on sentences that seem to require lookahead, such as, *I know that guy likes ice cream.*
>
> The new parser design does not recover tree structure; instead it builds up a modified *phrase marker* representation. This representational format directly encodes the co-identification of lexical and phrasal heads (following $\overline{\text{X}}$ theory). Interestingly, the representation is formally constrained in such a way that only certain grammatical operations can be stated, and these correspond to what is actually required in current theories. For instance, Chomsky adjunction, but not sister adjunction, is possible.
>
> The new parser design is completely modular. Information from a variety of sources may be folded together to yield the output representation, in a layer-cake fashion.

Syntactic ambiguity is easily encoded, with extra-syntactic resolution of ambiguity requiring minimal additions to the output representation.

To begin, we discuss the new parser design. We use roughly the same bounded-context design outlined by Marcus (1980) and modified in Berwick and Weinberg (1984). At any given point in the analysis of a sentence, the parser has access to certain bounded left- and right-context information; this is used to fix the parser's next move. The parser is deterministic, in a sense made precise below. The left-context represents the analysis of the sentence seen so far, while the right-context holds words. In the Berwick and Weinberg model, left-context could consist of at most a single constituent domain plus at least one addition cyclic node domain; within these constituent domains, only nodes c-commanding the current token scanned are accessible. Thus the left-context is literally bounded in this model. We shall adopt these restrictions here.

In the Marcus design, and in Berwick and Weinberg (1984), the right-context was taken to be at least the current word being analyzed plus (usually) two lookahead symbols. In the new model, there will be no lookahead; only the current token scanned is visible. In this way the model follows the Kashket and Abney designs. We could replace its explicit operations by licensing as well, but have not done so here.

The new parser will be deterministic, in the following sense. At any step i in a parse we denote the representation of the sentence analysis at that point by A_i. A parser will be said to be *deterministic* if and only if the sequence of sets A_1, A_2, ..., is monotonically increasing in an information theoretic sense: each set in the sequence is either the same as or is a refinement of its predecessor. This means that one cannot *remove* information from an analysis set once it is placed there; one may, of course, refine this information. We shall see that this constraint has important implications for what grammatical operations can and cannot be represented in the new parser design, as well as for the resolution of ambiguous sentence examples.

So far, aside from the lack of lookahead, the parser design we have sketched is identical to that in Berwick and Weinberg (1984). We now turn to the key difference between the two models. Instead of reconstructing explicit tree structure as a part of its left-context representation and as its final representation of sentences, the new parser will build as its output representation a *phrase marker* in the original sense used in transformational grammar: a set representing all and only the *is-a* and *left-to-right precedence* relations in a sentence. This representation is augmented to directly encode the key $\overline{X}$ constraint identifying heads of phrases with lexical heads and the indexing of, for example, traces. Given Kashket's design, it would probably be more correct to parcel out the levels of precedence and dominance into two distinct representations, but we will not do that here (we will deal only with English).

An example that will be used later on will help to pin this notation down and show its similarity to the usual CF parsing analysis. Suppose we have the sentence, *I know that block supports the pyramid*. First, number the positions between the word tokens:

```
0 I 1 know 2 that 3 block 4 supports 5 the 6 pyramid 7 • 8
```

Here is the output set for this sentence (using traditional *is-a* symbols for now). Each element of the set is a triple, consisting of first the phrasal name followed followed by its *start* and *stop* positions. (Nothing crucial hinges on the exact details of this phrasal analysis.)

```
{ (S 0 7) (NP 0 1) (PRONOUN 0 1) (INFL 1 1)
(V 1 2) (VP 1 7) (DET 2 3) (NP 2 4) (NBAR1 3 4) (NOUN1 3 4)
(S 2 7) (INFL 4 4) (V 4 5) (VP 4 7) (NP 5 7) (DET 5 6)
(NBAR 6 7) (NOUN 6 7) }
```

Note that this information will allow one to reconstruct hierarchical relationships among the elements, if desired, though remaining ambiguous in the case of nonbranching domination, since then we would have, e.g., `(NP 1 3)`, `(NP 1 3)`. (Indeed, in the output representation these two NPs would be merged and thus be quite literally indistinguishable.) [51]

Moving closer to our desired $\overline{\mathrm{X}}$ representation, let us now replace the conventional symbols NP and VP with their co-identified lexical heads. That is, let us say that the features of a maximal projection are coextensive with those of the lexical item that heads it. We indicate this by an index on a maximal projection identical to that of its lexical head. The index actually corresponds to a feature bundle, but is used simply as an abbreviation.[52]

{ (X-max i (=INFL) 0 7) (X-max j 0 1) (I j 0 1) (INFL i 1 1)

[51]This representation, in fact, is exactly that used by so-called "tabular" parsing methods, such as Earley's algorithm. At every step in a parse, we simply *assert* which phrasal analyses are compatible with the input seen so far. In this regard, the representational formal is quite like Kaplan's General Syntactic Processor (1973). Kaplan, however, aimed to build a tree once the assertions had been completely built. But this is unnecessary; indeed, as we shall see, it is the source of parsing difficulties, not a solution.

[52]The features ±*lexical*, ±*maximal* could be used to denote lexical (X^0) and maximal projections respectively; other combinations of these features yield intermediate bar levels.

(know k 1 2) (X-max k 1 7) (that 2 3) (X-max l 2 4) (NBAR1 3 4) (block l 3 4) (X-max m 2 7) (INFL m 4 4) (supports n 4 5) (X-max n 4 7) (X-max o 5 7) (the 5 6) (NBAR 6 7) (pyramid o 6 7) }

This is the *output* available at a parse's end. Of course, at intermediate stages of sentence analysis only intermediate portions of this set are available. The parser consults a bounded left-context representation of this set plus the current input token scanned in order to fix its next move. This context is placed on a pushdown stack in the manner described in Berwick and Weinberg (1984).

Note that at any stage the analysis is simply a *set* of assertions comprising *is-a* and left-to-right precedence relationships compatible with the input seen so far. In fact, this is entirely equivalent to the *prefixes* of phrase markers, in the traditional sense (only prefixes, of course, since the parser cannot tell us anything about input that it has not seen so far). By the end of the parse, the assertion set will simply be the phrase markers compatible with the input string.

Information recorded at every step in the parse will be monotonically preserved. Once a recorded decision about the *is-a* or start and end points of an element can no longer be refined, this commitment will not be revoked. That is, once we have fixed that an element of the assertion set is an NP, that cannot be changed to VP. Similarly, a decision that the NP extends between positions 1 and 3 of the input cannot be retracted once it is made. However, it is possible to *refine* a decision in an information theoretic sense, in two ways: first, one can add features to an is-a element, using the $\overline{\text{X}}$ features (e.g., changing an XP, with no features, to a +N item, to a +N −V item), provided that the element is still part of the left-context); second, we introduce a special asterisk notation, *, indicating that the *end* of a constituent has not yet been seen (this is apparently inevitable in a left-to-right analysis); third, we may *add* a new assertion, provided that the relevant portion of the phrase marker set is accessible. The asterisk, since it represents *no* information about the end position of a constituent, may be refined by replacing it with a specific numerical value, but of course, once this is done, the numerical value may not be retracted.

Unlike the Marcus machine, the parser will not use lookahead. Instead, its parsing decisions are based entirely on the current word token scanned and the representation of the input built up so far (the so-called left context of the parse). We may restrict the left-context, as is done in Berwick and Weinberg, to a literal encoding of what amounts to one S, but this won't be necessary in the sequel here. At each step in the parse, the assertion set of triples maintained will simply be the intersection of all triples compatible with the input seen so far. Thus at each step, the assertion set will encode just what is *know for certain* given the input so far processed. (This is a precise way to formulate the "wait and see" approach suggested by Marcus.)

Finally, the parser will be able to make top-down predictions (as in Marcus's parser) based on, e.g., subcategorization information, as when the lexical entry for a verb of a certain type sparks an expectation that a certain argument structure is expected. And, as in Marcus's design, the parser can certainly create assertions based on the actual words it finds in the input.

5.2.1 The $\overline{\mathrm{X}}$ parser in action

Now consider some examples that might seem to require lookahead. Note that our current design doesn't use lookahead at all—just the current token of the input. We shall show that using $\overline{\mathrm{X}}$ theory directly comes to the rescue—that a principle-based approach, using the right grammatical vocabulary, does better than an explicit rule system.

In English, words such as *for* and *that* are ambiguous. *For* can be either a preposition or a complementizer; *that* can be a pronoun, determiner, or complementizer. What must happen for a parser to successfully distinguish between these various uses?

> I know that block supports a pyramid. (*that* is a determiner)
>
> I know that blocks support the pyramid. (*that* is a Complementizer)

Examples such as these lead one to argue that one cannot parse deterministically without looking at one or more tokens ahead into the input, to see with *guy* is singular or plural.[53] If we cannot use lookahead, and we must decide whether to parse *that* as part of a Noun Phrase or a new Sentence, we are stuck. This is also true of our explicit rewrite rule approach, as used in the new parser design. If the only nonterminals names are objects such as Complementizer and NP, and if we must choose between these without retraction, then we are in trouble.[54]

Suppose, though, the the $\overline{\mathrm{X}}$ representational format is directly incorporated in the parser. Then parsing can proceed deterministically, or, rather, monotonically. In particular, suppose that we adopt Stowell's idea that a verb such as *know* subcategorizes a Case Recipient argument (NP or S); and that all Heads project to form phrases of some kind.

[53]One can construct real garden path examples in the worst case: *I know that blocks support for the pyramid is failing.*

[54]Indeed, the Marcus "diagnostic" rule to handle *that* is among the most baroque in his entire grammar: it specifically violates the general left-to-right operating principle of his machine, attaching the *first* item in the input buffer to the *second.*

Now let us see what happens when the parser analyzes a sentence such as, *I know that the block supports a pyramid.* We pick up the action at the time just after *know* is scanned. Since *know* predicts that a Case Recipient starts in the position just after it, we add the assertion triple (Case Recipient 2, 8) to the assertion set. (We might just use a diacritical x here instead of Case Recipient; this doesn't really matter.)

Now what? Reading *that*, the parser's compatible assertions include: (*that*, with no features, or perhaps +N, 2 3); (X-max 2 *) (projected from *that*, now assuming the $\overline{\mathrm{X}}$ theory); (Case Recipient 2 *); and the remainder from the earlier portions of the parse. Note that the features of the X-max, and *that* remain underdetermined; this is the best the machine can do at this point.

The parser now scans *block*. We add the triple (*block* +N −V, 3 4); but we can now also *refine* the earlier triples, (X-max, 2, *) to (X-max +N −V , 2 *). Finally, by scanning *support* we can then add whatever refinement we want to the Case Recipient triple, turning it into an S-bar (or S, as one might wish in some cases). Eventually, the X-max (NP) will span positions 2 through 4; the Case Recipient, 2 through 7.

Now let us now try *I know that blocks support the pyramid.* Again we predict a (Case Recipient, 2, *) triple after *know*, and (X-max 2 *); (*that*, 2 3). Now the parser sees a plural Noun, *blocks.* Based on this information, it can now refine the X-max triple to (X-max + whatever feature complex S-bars have, 2, *); (*that* + features for COMP, 2, 3); and (Case Recipient, + features for S-bar, 2, *). We also add the new projection (X-max +N −V, 3, *) (for the NP headed by *blocks*), and one for an S; note that this is always allowed. Eventually, the $\overline{\mathrm{S}}$ will stretch from position 2 through 7; the S, from position 3 through 7.

Note that no lookahead is required to resolve these cases, nor any abnormal rule that attaches items in a peculiar fashion. The "wait and see" or "attention shifting" behavior of the Marcus parser does not need to be stipulated; rather, it falls out of the normal behavior of the parser, operating with $\overline{\mathrm{X}}$ principles and assertions. Where the $\overline{\mathrm{X}}$ system underdetermines feature complexes, we get the *effect* of wait-and-see behavior, but it is part and parcel of the normal operation of the machine—nothing special at all need be said in order to get this effect.

The same approach resolves many *for* or *to* ambiguities.[55] If the grammar has to use the categories PP, $\overline{\mathrm{S}}$, and so forth, then our deterministic parser must fix this categorial status correctly, without extra lookahead in the proposed model. But then the parser cannot correctly decide between *for* as the Head of a Prepositional Phrase and *for* as the

[55]It is interesting to speculate as to the existence of function word homonyms of this kind; are these accidental or not?

a Complementizer—the Head of an $\bar{S}$, in some theories. Now suppose that the $\bar{X}$ schema is directly used by the parser. Then, given that *for* is a Head of some kind, the parser can create a maximal projection, with a kind of "operator" status (the Head element). The features of this projection can remain partially determined until the following material is analyzed as either an NP followed by verbal material (then *for* heads an $\bar{S}$) or just an NP (*for* heads a PP). In fact, on this alternative, there is no "PP" or "$\bar{S}$" at all, just the Head with whatever features it has.

Next consider *to* ambiguities. No matter whether *to* is a Preposition or Infl, it is a case-marking operator; this is what the underdetermined $\bar{X}$ machinery projects. The same holds for *have* considered as either an auxiliary verb (Infl) or a main verb. In either case, *have* forms a maximal +V projection. One can delay a decision to insert an NP in a sentence such as *Have the students take the exam* until *after* the NP is processed, since this will involve simply *adding* a new assertion, not destroying any old assertions. Thus no lookahead is required in this case either.

Let's take a more complicated example. Consider another sentence type where a verb can subcategorize for either NP or $\bar{S}$, e.g, *I expect John/I expect John to leave.* Exactly the same approach will work here as before. In the first sentence, the X-Case Recipient triple will eventually be co-identified with the N-max projection, via refinement. In the second, the X-Case Recipient will be refined to be an S-bar spanning positions 2 to 5, while the N-max will stretch only from position 2 to position 3 in the input. Refinement occurs automatically as each token is scanned: when the infinitive *to* is reached, the N-max is closed off at position 3 and its N-max features not percolated to the X-Case Recipient. No lookahead is required.

At the same time, because no lookahead is ever used, "short" garden paths, such as *the boat floated sank* are at least possible to envision; the Marcus parser had to deliberately ignore its own lookahead information in an ad hoc way in order to garden path in such cases. Note that this is in contrast to the ambiguous subcategorization of *know*, where we had a particular alternative that could later be refined. In the present sentence, there is no obligatory argument after *floated*, so none can be safely posited. Unless extra information is provided that sets up an alternative that may safely predict an operator, then the parse will fail. This would be true, for example, if *that* were present. But now note that we have reconstructed—in a *principled* way—the canonical surface sentoid strategy of Fodor, Bever, and Garrett (1974): "take N-V as a main sentence unless there is a surface mark of embedding."

5.2.2 Imposing constraints in the $\overline{\text{X}}$ parser

The set-based model lends itself to a rather different picture of parsing, perhaps best dubbed "layer-cake parsing": each layer is an additional constraint set to be imposed on the initial analysis provided by the parser. These constraints might include lexical, intonational, word frequency effects, or even (after syntactic processing is complete) plausibility filtering. A modular view makes this possible.

To see how this might work, we will draw on some examples from Berwick and Barton (1985) and Berwick and Weinberg (1985):[56] one dealing with PP attachment ambiguities, and one with so-called readjustment rules (Langendoen 1975). Our point is simply that the direct use of $\overline{\text{X}}$ rules in a set format makes adding constraints easy.

Consider first the typical PP ambiguity sentences:

$_0$ I $_1$ saw $_2$ the $_3$ guy $_4$ on $_5$ the $_6$ hill $_7$ with $_8$ the $_9$ telescope $_{10}$

What can the parser say for certain here? The individual NP and PP phrases are positively known:

(NP 2 4)	(PP 4 7)	(NP 5 7)
(PP 7 10)	(NP 8 10)	

This base can be easily augmented to incorporate lexical or intonational effects—the point being that all that we need are monotonic refinements. For example, if we add the assertions (NP 2 10) and (NP 5 10) then we get a completely right-branching analysis. Adding just (NP 5 10) gives us the reading where *on the hill with the telescope* modifies *see*; adding just (NP 2 7) gives us [*guy on the hill*][*with the telescope*], and so forth.

Intonational effects may be easily incorporated as obligatory constraints laid on top of the base set. Suppose that clause-final lengthening and onset stress provide the cues for clause boundaries (XPs) but nothing more:[57]

I saw the guy on the hill ↓ **with** the telescope.

[56]The set-based representation also bears a close relationship to the proposals of Marcus regarding a representation using just the predicates *precedes* and *dominates*. The similarities are plain enough, so they won't be spelled out here.

[57]Matters become more complex if we would have to introduce certainty factors into this information, but this could be done as well.

Then we would expect the intonational component to propose the following assertions: (X^{max} 2 7) (X^{max} 7 10). We may now impose this constraint on our base set, yielding (NP 2 4), (PP 4 6), and (X^{max} 2 6). Since every maximal projection must be co-identified with some lexical head, our remaining choice is to make the X^{max} an NP or a PP. A head-first constraint makes it the NP.

A similar analysis works well for proposed cases of so-called "readjustment rules," but in this case we can again give a *principled* account for them—that is, they are a side-effect of our more basic $\overline{X}$ parser. Consider the classic Chomsky-Miller example where intonational contours run counter to syntactic contours: *this is the cat that chased the rat that ate the cheese*. Suppose we take intonations as providing us with the following surface analysis of such sentences, with extraposition of relative clauses:

[the cat$_j$ t_i] [that t_j chased the rat$_k$ t_l]$_i$ [that t_k ate the cheese]$_l$

From this we can construct the following assertions (where an empty category is in the form (NP x x), dominating no phonological material in the string):

(NP$_j$ 0 2)	(Det 0 1)	(N$_j$ 1 2)
($\overline{S}_i$ 2 2)	(Sbar$_i$ 2 6)	(that 2 3)
(NP$_j$ 3 3)	(VP 3 6)	(V 3 4)
(NP$_k$ 4 6)	(Det 4 5)	(N$_k$ 5 6)
($\overline{S}_l$ 6 6)	($\overline{S}_l$ 6 10)	(that 6 7)
(NP$_k$ 7 7)	(VP 7 10)	(V 7 8)
(NP$_m$ 8 10)	(Det 8 9)	(N$_m$ 9 10)

The intonational evidence is compatible with the assertions as listed; for example, there is an XP between positions 0 and 2, and 6–10. More importantly, in order to "readjust" the structure to reconstruct the right S-structure one only needs to *add* certain statements, such as (NP 0 6), as indicated by the indices. No special readjustment rules of any sort are called for.

5.2.3 Summary of $\overline{X}$ parsing

In short, the $\overline{X}$ parser, coupled with principled assertions in a "layer cake" model, work well together and better than simple context-free rules to accumulate evidence in a monotonic way for parsing. The most important moral is that all this works naturally *if* one is willing to parse using $\overline{X}$ principles directly, rather than indirectly. In addition, the framework sketched here is compatible with constraints that allow the order among complements to a verb, to be free, if there is no other constraint on them. That is, in order to establish what constituents are to be expected after a verb, we simply project, directly, the lexical subcategorization frame of the verb as it is found; there is no need for an ordered righthand side as in a traditional context-free rule. Whatever assertions are compatible with the (unordered) subcategorization frame will simply be made as before. Of course, if there *are* constraints, such as case assignment/adjacency considerations, these may be enforced and the resulting assertion sets filtered; no change in the basic parsing machinery is required.

To summarize: in this section we've presented an $\overline{X}$ parser that uses underdetermined features in order to monotonically analyze sentence structure. The essential idea is that many locally ambiguous parses have the same topological structure, so we can force the parse forward, deterministically, until resolving lexical information is found. The parser is more "compiled" than the Dorr or Kashket designs, since it projects categorial and, if need be, Case features; yet it is still firmly wedded to $\overline{X}$ principles.[58]

[58]It may even suggest why sentences such as *The students knew the solution to the problem would be easy* might result in an additional processing load as compared to sentences such as *the students knew the solution to the problem*: the parser has a bigger refinement job (adding a new element rather than just refining one) in the first kind of sentence. But these details remain to be worked out.

6 Principle-based translation

This section aims to demonstrate the advantages of principle-based parsing in a different domain: syntactic language translation.[59] We describe a particular principle-based translation system, implemented by B. Dorr at MIT, and show how it successfully overcomes the difficulties of a rule-based approach. In particular, it avoids the large, highly idiosyncratic grammars so characteristic of context-free based syntactic translators. As Dorr (1986:6) puts it:

> The reason that translation is difficult is [in part] that it seems to require a massive amount of "knowledge" in order to handle all possible phenomena (and their interaction effects) that might occur in a language. Consider (1)
>
> (1) ¿Qué vio?
> 'What did {he,she} see?'[60]
>
> Although (1) appears to be simple, it is not simple from the point of view of machine translation since the sentence exhibits interaction among three complex phenomena. The first phenomenon is the absence of a subject in the source language. In many languages (e.g., Spanish, Italian, Greek, and Hebrew), the pronominal subject of a tensed sentence may remain unexpressed; the verbal morphology is rich enough to make the subject pronouns recoverable. Thus, (1) would literally translate as:
>
> (2) * What saw?[61]
>
> In order to rule out (2) a translation system must have the "knowledge" that a null subject is allowed in Spanish, but not in English.
>
> The second phenomenon concerns movement of a sentence constituent. In (1), the verb *vio* (= saw) takes an object, but the object does not appear where it is normally found (i.e., after the verb); instead, the object *qué* precedes the verb. The positioning of *qué* at the beginning of the sentence is conceived of as a type of movement: the object has moved from verb-phrase final to sentence-initial position. This phenomenon, which may occur in either Spanish or English, must be accounted for during the translation process.

[59] By this we mean that aside from considerations of thematic role matching, no "semantic" considerations enter into the analysis.

[60] The "{.. , ..}" notation will be used to denote optionality. Thus, in (1) the subject of the sentence may either be *he* or *she*.

[61] An asterisk in front of a sentence is used to indicate that it is syntactically ill-formed in some way.

The third phenomenon is called inversion. In Spanish (and other Romance languages), the verb is allowed to invert with its subject. Thus, if the subject of (1) were overt, it would be inverted, as in (3):

(3) ¿Qué vio Juan?
'What did John see?'

Although the subject is phonetically null in (1), it is assumed to have inverted with the verb, just as in (3), where the subject is present.

Taken together with the rest of the surface constructions in a particular language, such effects would demand a large number of individual context-free rules. As a result, most such systems have had to limit their linguistic coverage. To demonstrate this, we will follow Dorr (1986) and first describe what a typical rule-based translation system looks like, taking as a concrete example Slocum's METAL system, a Spanish-English translator. We then turn to Dorr's Spanish-English principle-based design.[62] Dorr's translator is interesting not only because it can actually handle sentences like *¿Qué vio Juan?*, but also because its parser uses a novel co-routined design: essentially pure $\overline{X}$ skeletal parsing, operating in tandem with GB constraints.

6.1 Context-free, rule-based translation: METAL

We begin with an example of a rule-based translation system, METAL (Slocum 1984). METAL has a separate set of rules to translate between, say, German and Spanish and German and Chinese; this is called a "transfer" approach in the literature. There is no rule sharing; all rules are considered idiosyncratic and language-particular. See figure 16.

Dorr describes a sample METAL context-free rule and its difficulties as follows:

Figure 17 gives an example of a context-free rule in the METAL system. In this example, the "father" node (NN) and "sons" (NST and N-FLEX) are built into a syntax tree as shown in figure 18.[63] This tree corresponds to a noun stem (NST) and its nominal ending (N-FLEX). The syntax tree is built by the CONSTRuctor part of the rule only after the tests in the column tests (the second and third lines) and the TEST portion of the rule are satisfied. Essentially, what this rule does is associate a noun with with the two constituents NST and N-FLEX (using the column tests which require the first element to be word-initial (WI) and the second element to be word-final (WF)), and then test for

[62]Sharp (1985) has also built a GB-based Prolog Spanish-English translator that differs sharply from Dorr's design in that it multiplies out many more principles. We discuss this point in more detail below.
[63]This example is taken from Slocum, 1984:18.

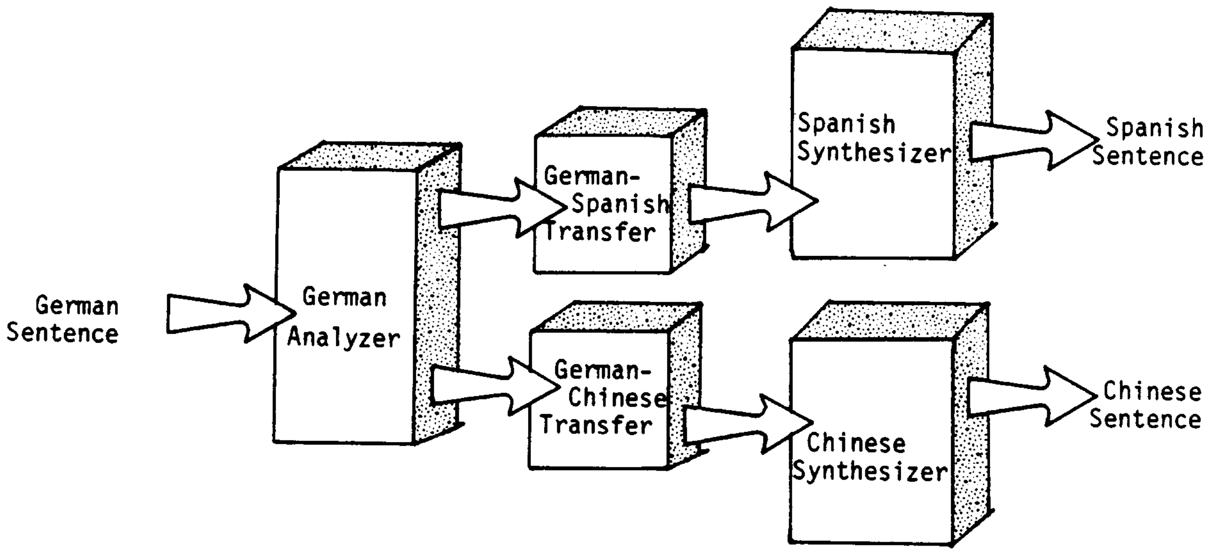


Figure 16: Transfer Translation Approach as found in METAL (1984) (taken from Dorr 1986)

agreement between the two constituents (the fourth line). Note that the application of such a rule must be restricted by tests on syntax (positioning and agreement in this example) which are encoded directly in the rule, rather than in independent modules which can account for these constraints globally. Furthermore, in order to account for a wide range of phenomena, thousands of such rules are required for each language, thus increasing grammar search time.[64]

6.2 Principle-based translation: UNITRAN

Dorr's translation scheme is interlingual (see figure 19); we focus on the analyzer component here, since it is a principle-based parser.

Dorr's parser works by using the Earley parser on very slightly expanded $\overline{X}$ skeletons. These skeletons guide an Earley-algorithm parser, that works in tandem with GB principles. By modularizing the principles in this way, subcategorization information is *not* multiplied into a rule system; rather, it is accessed on-line as it is needed.

[64]The LRC MT system has approximately 1,000 phrase-structure rules for each source language.

```
 NN          NST       N-FLEX
  0           1          2
LVL 0       REQ WI     REQ WF

TEST (INT 1 CL 2 CL)

CONSTR (CPX 2 ALO CL)
       (CPY 2 NU CA)
       (CPY 1 WI)
```

Figure 17: Context-free Phrase-Structure Rule in METAL (taken from Dorr 1986)

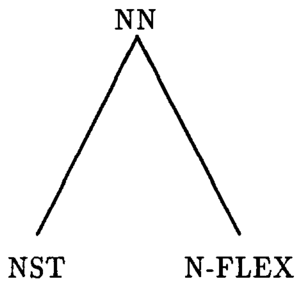


Figure 18: Tree generated via Context-free Phrase-Structure Rule in METAL (taken from Dorr 1986)

Dorr assumes that $\overline{\mathrm{X}}$ theory provides the basic phrase configuration possibilities (across all languages). The basic rule

(4) $\overline{\overline{\mathrm{X}}} \Rightarrow$ Specifier $\overline{\mathrm{X}}$ Complement

is combined with a rule to handle adjuncts

(5) $\overline{\mathrm{X}} \Rightarrow$ Adjunct $\overline{\mathrm{X}}$
$\overline{\mathrm{X}} \Rightarrow \overline{\mathrm{X}}$ Adjunct

to yield the following schema:

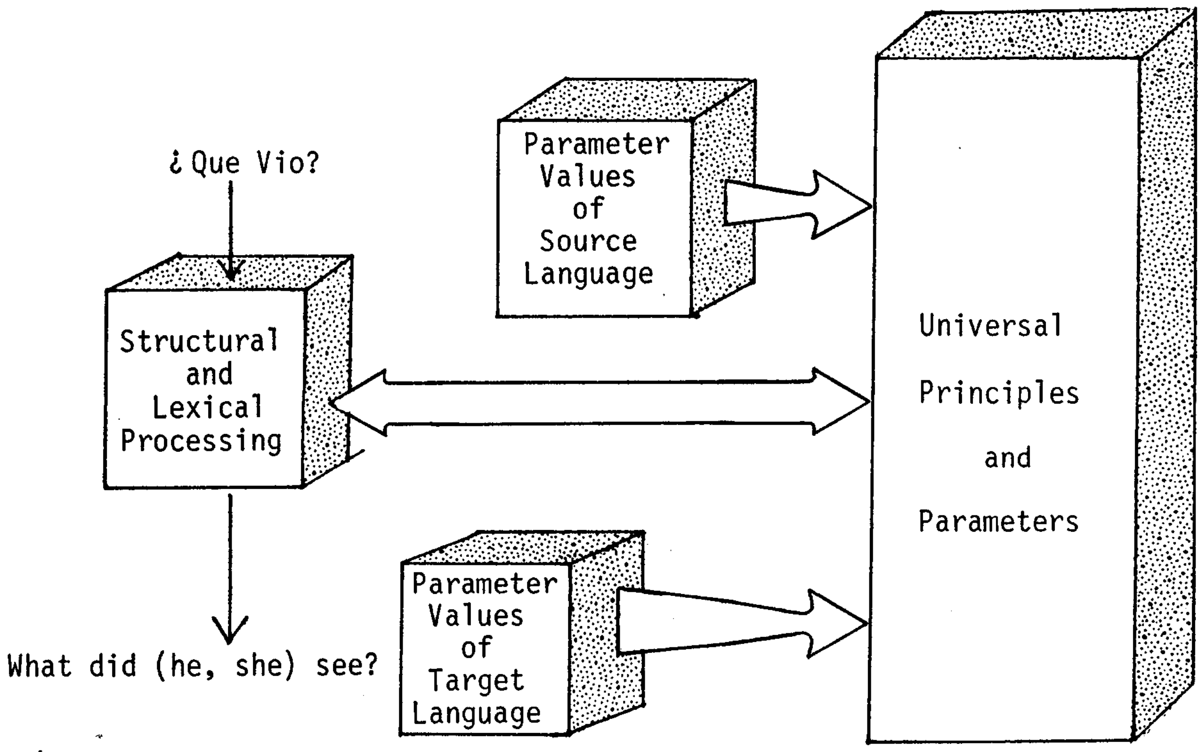

¿Que Vio?
Structural and Lexical Processing
Parameter Values of Source Language
Universal Principles and Parameters
What did (he, she) see?
Parameter Values of Target Language

Figure 19: The Modified Interlingual Design: Dorr 1986

(6) $\bar{\bar{X}} \Rightarrow$ Specifier $\bar{X}$ Complement
$\bar{X} \Rightarrow$ Adjunct $\bar{X}$
$\bar{X} \Rightarrow \bar{X}$ Adjunct
$\bar{X} \Rightarrow$ X

In addition, Dorr sets a parameter so that adjuncts may occur before or after the specifier and before or after the complement (in a specifier-head-complement language). Finally, if we vary the order of specifier and complement with respect to the head, we have the 3!=6 possible tree topologies shown in figure 20, where (a) corresponds, for example, to English phrase structure order. What Dorr has done, then, is to "compile out" information about skeletal phrase structure possibilities only mildly; note that these skeletons do provide topological information for parsing but do *not* provide any detail about categorial identity or subcategorization information down to the level of individual subcategorization frames. (However, the system can access online lexical category information—whether a word is a Noun or a Verb, or its binary feature equivalents. It also knows about a few other

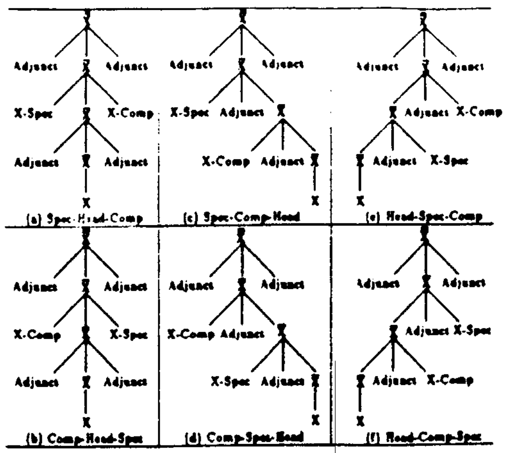


Figure 20: Dorr's $\overline{\text{X}}$ theory provides for six basic $\overline{\text{X}}$ skeletons.

parameters in each particular language, and multiplies these into the $\overline{\text{X}}$ skeletons: choice of specifier, what a possible trace can be in a particular language, and what a complement might be if a COMP or INFL head is missing.)

To summarize, the following information is precompiled in Dorr's system (note that the trace module is partially accessed to find out what possible traces there may be in a pro-drop language):

The $\overline{\text{X}}$ order for a language;

Specifier choice (e.g., det for $\overline{\overline{\text{N}}}$ in Spanish)

The adjunction possibilities (e.g., clitic to $\overline{\text{V}}$ in Spanish)

Default values for nonlexical heads (e.g., $\overline{\overline{\text{V}}}$ complement for I(nflection) in Spanish)

To get the actual context-free rules for Spanish, we simply instantiate template (6)-(a) with values for X={N, V, P, A } *at the time aprsing occurs*, i.e., on-line, plus two rules for INFL and COMP (also ternary, following details that are irrelevant here). For Spanish, the system knows that the choice of specifier for COMP″ may be a wh-phrase; and it also knows that if a COMP head is absent, then the complement of COMP is INFL″. Finally, it knows that V and N″ are adjunction possibilities. Note that these rules do not form a complete description of Spanish, and are not intended to: they are indeed underconstrained. The entire $\overline{\text{X}}$ skeleton system multiplies out to on the order of a hundred or so context-free rules.

Dorr's full parser works by using the $\overline{\text{X}}$ skeleton rules as a driver program coroutined with GB principles, as sketched in figure 21. The $\overline{\text{X}}$ component vastly overgenerates; it builds underspecified phrase structure because it does not access subcategorization details. The actual parser is simply an Earley parser, using the context-free rules given in (6). As each word is processed, it operates using the PUSH, SCAN, or POP actions of the Earley algorithm until no more actions trigger; it then calls on the GB component. The GB module also calls on the PUSH, SCAN, and POP actions, but in a much more complicated way. In effect, the skeletal parser has only a vague idea of the actual structure of sentences, which is just used just to keep the Earley algorithm's PUSH-SCAN-POP sequence going. By splitting up the computational work in this way, we can vastly reduce the size of the context-free component needed, because we do not multiply out the rules used.

The GB component has three tasks, accessed on demand: first, it weeds out bad parses (for example, a parse that calls for a complement when the subcategorization frame does not need one or if bounding conditions are violated); second, it tries out possibilities that the $\overline{\text{X}}$ skeletal parser does not know about (for example, it can PREDICT that an empty category must be present at a certain point); and third it tries to extend the Earley parse to a point where skeletal parsing can take over again (for example, by assigning and checking thematic roles, it can get the parse to the point where a phrase is complete, and so the Earley algorithm can execute a POP action).[65] Figure 22 summarizes the jobs the $\overline{\text{X}}$ and GB modules have. The overall effect is to implement and back-and-forth flow of information between the Earley parser actions and the GB filtering constraints and actions. (All parsing possibilities are carried along in parallel, as is typical with the Earley algorithm.)

For each particular language, there will also be a set of parameterized choices in the $\overline{\text{X}}$ and GB modules. For example, Spanish permits $\overline{\overline{\text{N}}}$, wh-phrase, V, Infl, have-aux, and be-aux as traces; while English permits only $\overline{\overline{\text{N}}}$and wh-phrase. These are used for predicting a trace at PUSH time. Similarly, the governing nodes and proper governors may differ between the two languages: these are V, N, A, P, AGReement in both languages, while Dorr considers the proper governors to be V, INFL in Spanish and V, P in English. These are used to check well-formedness at PUSH/POP time.

To see how this all fits together, consider the simple sentence *¿Qué vio ?* ("What did she, he see"). Figures 23 and 24 show the parsing sequence and final results. As mentioned, this sentence exhibits three interesting syntactic phenomena: a null subject; inversion of

[65]In most cases, the GB component will augment or eliminate trees. Some of the effects on the timing of the algorithm are discussed briefly below.

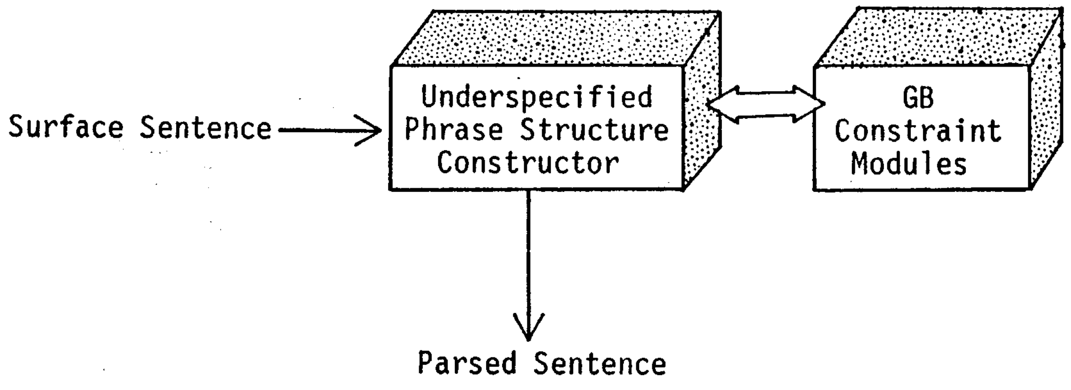


Figure 21: Back and forth flow of phrase structure constructor with GB constraint modules (taken from Dorr 1986)

the verb; and movement of *qué* to the front of the sentence.

Morphological analysis first reduces this to the actual parsed form *¿'Qué ver*+features past, singular, third person (with the root form for the verb, as is typical).

The parser first accesses the $\overline{X}$ skeleton for $\overline{\overline{\text{COMP}}}$, which is simply COMP-specifier–COMP–COMP-complement (in Spanish), and where Comp need not be overt. A precompiled parameter particular for Spanish forces selection of $\overline{\overline{\text{INFL}}}$ as the complement of COMP, since indeed there is no head present. *Qué* is then SCANned (attached) to $\overline{\overline{\text{COMP}}}$ as its (optional) specifier (note once again that this may lead to overgeneration since no conditions are checked at this point to determine whether this attachment is in fact correct).

Next, the expansion of $\overline{\overline{\text{INFL}}}$ has two $\overline{X}$ precompiled possibilities, both of which are pursued by the Earley algorithm.

First, INFL-spec INFL INFL-comp may be expanded as V–$\overline{\text{COMP}}$ (with V filling the spec slot of $\overline{\overline{\text{INFL}}}$); let us continue to call this parse (a). (Here, V is one of the adjunction possibilities that has been compiled out beforehand; adjunction is assumed to occur at the $\overline{X}$ level.) Second, INFL-spec may be expanded as $\overline{\overline{N}}$ and INFL-comp as $\overline{\overline{V}}$–V (parse c, step 2). At this point, the parser has access to the next word, *ver*, a verb. This rules out any expansion of $\overline{\overline{N}}$ as its usual spec-head-comp sequence. For this parse then, no further $\overline{X}$ actions can occur and the parser consults the GB module.

The GB module determines that the next symbol to operate on for parse (c) is a nonterminal; hence, a PUSH is required. The corresponding GB action is to predict an empty category: either a trace or pro. Both are predicted: there is a possible antecedent

ACTION	EARLEY PARSER	GB MODULE
PUSH	Expand nonterminal	Predict empty category
		Check trace and bounding
SCAN	Traverse terminal	Determine argument structure
		Perform feature matching
POP	Complete nonterminal	Assign thematic roles
		Check theta criterion

Figure 22: The Earley parser works in tandem with GB constraints in Dorr's design

that is not too far away, and the subject position (first $\overline{\overline{N}}$ under the INFL projection) is still open. (Several principles come into play here, notably bounding, proper government, and so on; each is checked separately on-line.) We will call these two ongoing possibilities parses (2c) and (2d). (For parse (2c), the binding module links the trace to *qué*; for parse (2d), the system knows that only small pro, not PRO, can be placed in the relevant position.) The GB module has nothing to say about parse possibility 1, and parsing is returned to the $\overline{X}$ component.

Finally, all in step 2, the precompiled $\overline{X}$ skeleton notes that INFL is currently absent, and so adds the (precompiled) complement possibility of $\overline{\overline{V}}$–V for parses (2c) and (2d).

We proceed to step 3. The parser next SCANs *ver*. It can be attached to the V slot for parse (3a). The GB module is then called upon: its job is to determine the argument structure for the verb *ver*. The lexical entry for *ver* predicts an $\overline{\overline{N}}$ complement.[66] For parses (c) and (d) this prediction may be realized only as an empty category trace, linked to the antecedent empty category (either trace or pro); for parse (a), however, the subject position, known to be (precompiled as) $\overline{\overline{N}}$, is still open, so either trace or (little) pro may be predicted here, yielding parses (4a) and a new parse (4b). In all, there are now four parses. (In so doing the parser rules out many possible realizations of the spec position for INFL, but we will not cover these here.)

[66]This is actually done in a more principled way, by projection from lexical-conceptual structure, but we ignore this detail here.

All words are now scanned, so the GB module is now called on to check for any additional actions (step 4). For parses (c) and (d), all that remains is thematic checking: the Theta Condition says that each arguments gets one and only one thematic role. However, both parses (c) and (d) violate this condition, since two semantic roles are assigned to the object position: object is both Goal and also, via its antecedent linking, as Agent. Thus these parses are thrown out.

For parses (a) and (b), additional actions are possible (steps 4–7). The complement of $\overline{\text{COMP}}$ is precompiled as a $\overline{\overline{\text{V}}}$ (as dictated by a precompiled parameter, not a rule, since INFL is empty) (step 4). Since there is no word present, this $\overline{\overline{\text{V}}}$ may be expanded (in Spanish) by a precompiled $\overline{\text{X}}$ template as an empty Verb (since verbs may be traces), linked to *ver* plus (via online accessed subcategorization facts because of linking to *ver*) an $\overline{\overline{\text{N}}}$(steps 5 and 6, parses (a) and (b)). Finally, the GB module must again be accessed at this point, since we have just PREDICTed as $\overline{\overline{\text{N}}}$. The $\overline{\overline{\text{N}}}$, by GB principles, may be realized only as a trace in this position, and this is what we get (see figure 24, step 7(b)). Transliterated, parse (a) means "What saw"; parse (b), "What did she, he see"; parse (c), "What did what see itself"; and parse (d), "What did she, he see himself, herself".

Note that of the remaining two possible parses, (a) and (b), (a) is ruled out by GB feature checking, since *ver* demands an animate subject. This leaves only parse (b), "What did she, he see" as valid (with the null subject parameter in Spanish permitting an empty category in subject position). Note that the three syntactic phenomena of null subject, inversion, and movement are covered by GB filtering principles guiding the $\overline{\text{X}}$ parse, rather than by particular rules.[67] Importantly, this is done by using about a hundred underspecified $\overline{\text{X}}$ skeletons that do not access subcategorization information until it is encountered (as in the Berwick and Barton parser of the previous section) plus a dozen or so GB principles.

To summarize, Dorr's parser is a particularly novel design that reveals how information clustering and compilation work together to make parsing more efficient. Her design uses underspecified $\overline{\text{X}}$ schemas, with partially multiplied-in information about traces and adjuncts, but without subcategorization information (thus answering some of Frazier's psycholinguistic objections to principle-based parsers). As figure 2.2.1 indicated, there seems to be an optimal balance between compiling out $\overline{\text{X}}$ rules and leaving some principles alone. At the same time, the co-routine design leads to a complex *surface* behavior that could lead one to believe that principles and $\overline{\text{X}}$ skeletons had been multiplied out together. In short: the Dorr design shows that we have just begun to scratch the surface for principle-based parser designs, and that psycholinguistic theories have a vast space of possibilities to explore.

[67] In fact, parse (a) would be ruled out sooner by the animacy condition, but we will put aside this detail here.

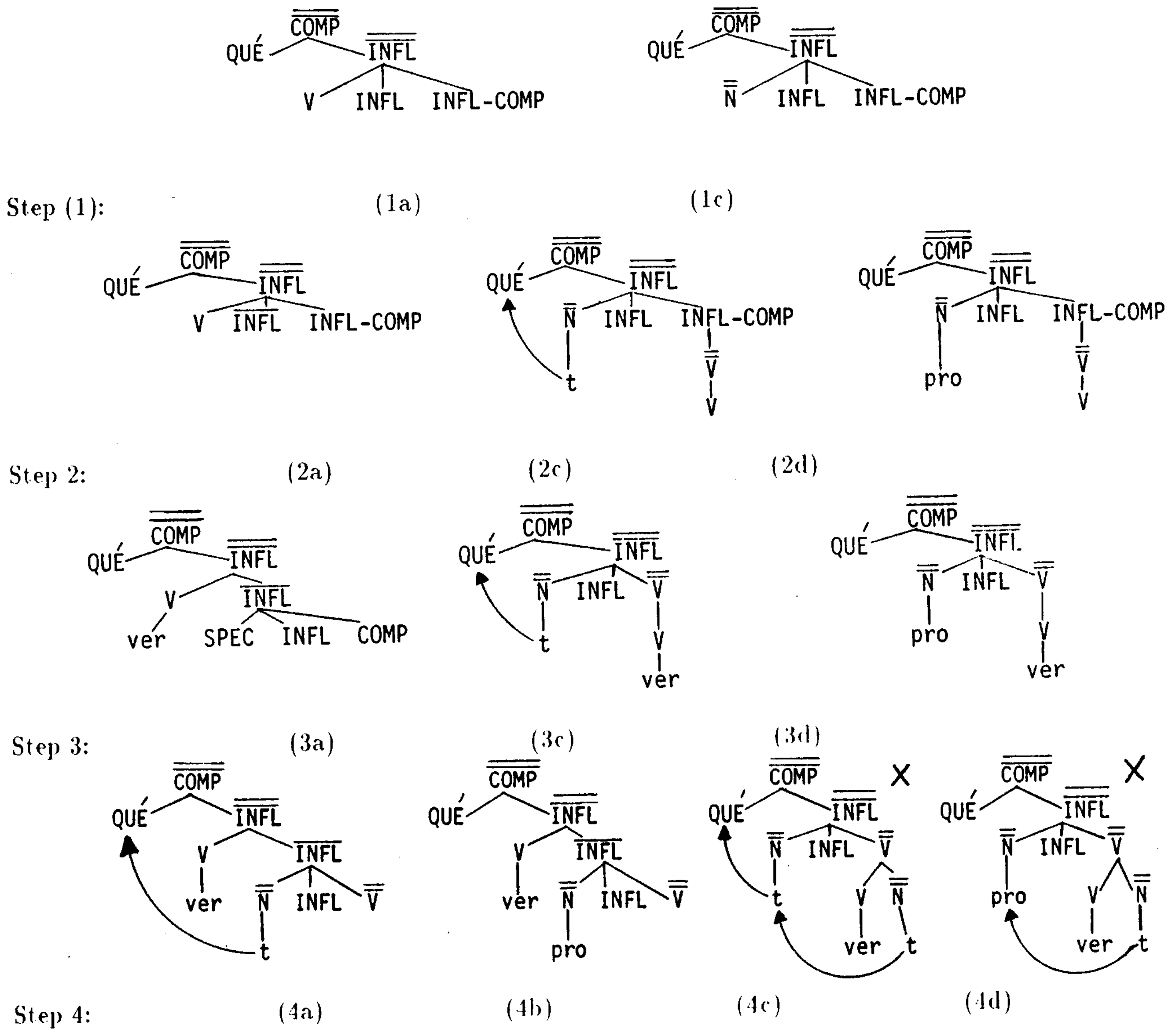


Figure 23: A simple Spanish sentence has four possible parses, but only one is valid. (Steps 5–7 are continued on the next page)

¿Qué vio ?

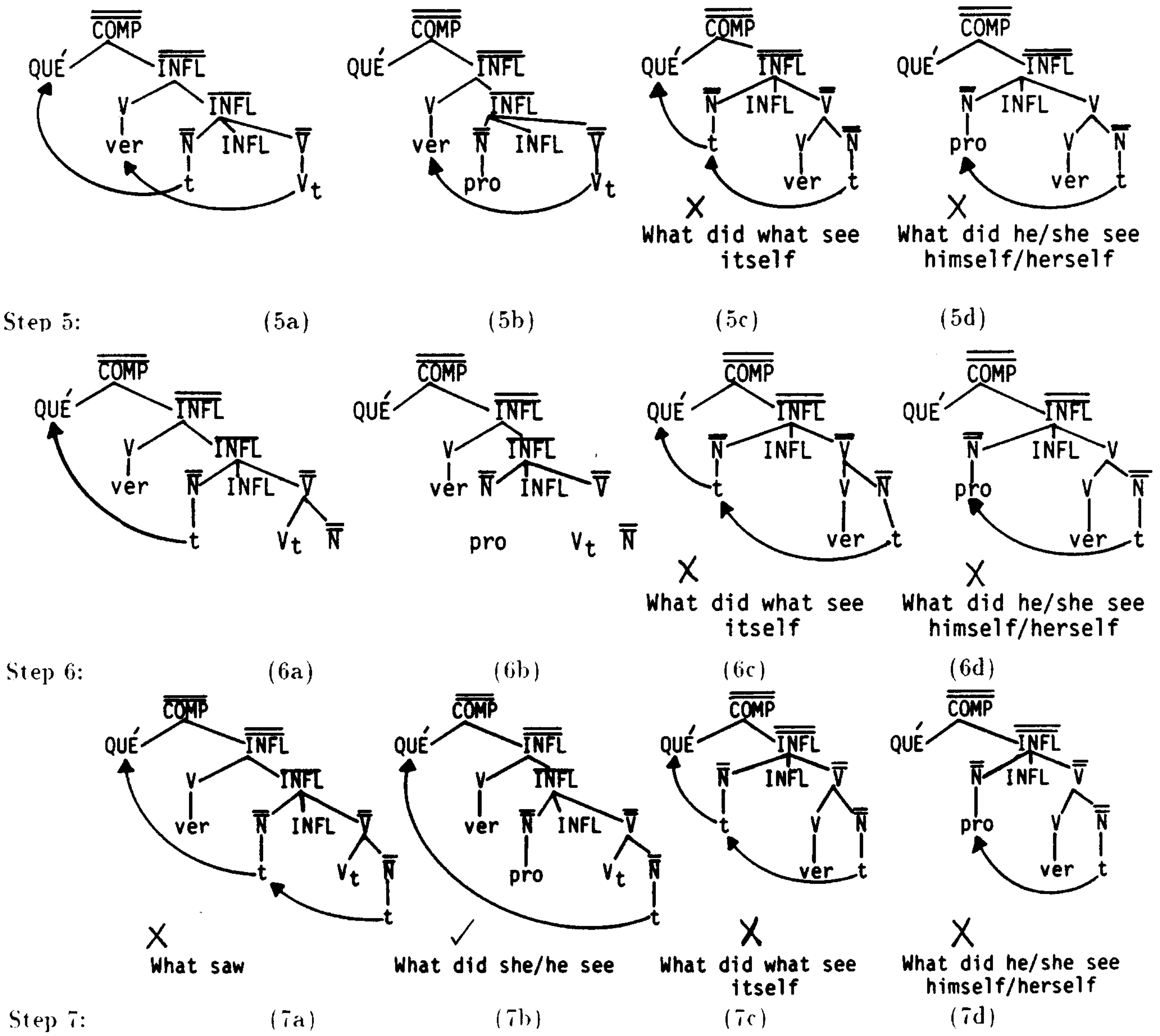


Figure 24: A simple Spanish sentence has four possible parses, but only one is valid.

7 Conclusion: modularity and principle-based parsing

The previous sections have given several working examples of principle-based parsers. One thing almost all have in common is modularity: they divide up grammatical representations into different vocabularies, parceling out the computational work that must be done into several different formats instead a single, monolithic representation.

One question that deserves to be raised pertains to the computational efficiency of principle-based approaches. While the implemented parsers justify themselves on this score—they seem about as efficient as large CF grammars, or the Marcus parser, hazards about such comparisons aside—one might wonder whether such modularity makes sense on more general grounds. In particular, we've seen that principle-based theories often posit a many-step derivation between principles and surface sentence; and this might take an equally long time for a parser to unwind.

To conclude this survey of principle-based parsing, I would like to speculate a bit on why in fact this might *not* be so—why, in fact, that modularity might be more efficient than a monolithic representational format.

There is one simple point to make and one deeper, relatively newer speculation. The simple point is that modularity makes for smaller grammars: smaller, because we can replace a multiplied-out rule set with an additive principle set. I won't delve into this obvious point very deeply. It has an interesting formal language theory backing; see, for example, the work of Platek and Sgall 1978 who show how a cascade of linear pushdown stack transductions of simple languages can give us fairly rich strictly context-sensitive languages—but not all context-sensitive languages—out the other end. This argues for a stratified, multiple-level linguistic theory. This formal point is echoed the parser construction business; one need only take a look at a worked-out example of how large a multiplied out rule set can become.[68]

The deeper point centers on the power of modularity used in conjunction with parallel processing. Roughly speaking, if a problem can be written down as a highly modular, planar graph, it can be solved by super-fast parallel circuits. If this property holds of subportions of a modular linguistic theory, then we might expect parallel speedup advantages here also by using so-called *constraint propagation* methods. It is not clear whether these properties hold of modern, principle-based theories of whatever model; however, it is at least clear that in certain domains, such as autosegmental theory, or in the two-level version of logical form proposed by Hornstein (1986) they do hold, and so we can expect

[68]See Ristad 1986, Appendix A and B; Barton, Berwick, and Ristad 1987 Appendix B for GPSG examples, including citations from working GPSG systems. But the same would be true for a multiplied-out GB-based parser.

significant parallel speedup, at least in principle.

Barton, Berwick, and Ristad (1987) summarize the relationship between modularity and parallel speedup this way:

> Intuitively, the easier it is to divide a graph into distinct and roughly equal pieces, the easier it is to use more efficient divide-and-conquer algorithms instead of combinatorial backtracking. A graph's *separability* tells us how many vertices we have to remove in order to obtain two roughly equal sets of nodes with no connections between them. For example, trees are highly separable; by removing a single vertex we can divide any n-vertex tree into two parts, each with no more than $2n/3$ vertices. Therefore, we would expect divide-and-conquer methods to be especially efficient on tree structures. ...
>
> Divide-and-conquer algorithms carve the original problem into two or more smaller problems. The subproblems are then solved by applying the divide-and-conquer algorithm recursively; the solutions to the subproblems are then combined to form the solution to the original problem. Divide-and-conquer methods work best when the subproblems are significantly smaller than the original problem.
>
> Similarly, certain separable 3SAT problems can be solved more quickly than general 3SAT problems, though the algorithms still take exponential time. In recent work, Kasif, Reif, and Sherlekar (1986) show that planar 3SAT problems—roughly, the class of 3SAT problems representable in a plane so that truth assignments don't cross—can be solved in serial time $O(2^{c\sqrt{n}})$ by a divide-and-conquer algorithm. This is still not polynomial time, but is much better than "brute force" backtracking with its $O(2^n)$ complexity. (On a parallel random access machine, they show this approach takes time $O(\log^3 n)$ using $O(n^2 \cdot 2^{O(\sqrt{n})})$ processors.)
>
> When the modularity of a 3SAT problem is even stronger, so that we can split a graphical 3SAT representation into about two equal pieces by removing only a *constant* number of vertices, then we can do even better: these problems are in $\mathcal{NC}$ and so can be solved by superfast parallel algorithms (taking only logarithmic time). (Kasif, Reif, and Sherlekar 1986).
>
> These examples are linguistically relevant. Chapter 6 develops an account of morphological analysis that relies on the algorithm known as constraint propagation. Constraint propagation is usually fast, but it can compute only local satisfiability constraints based on consistency between adjacent elements, not the global consistency and satisfaction properties ...

What is more interesting, however, is that while constraint propagation cannot always give correct answers to NP-hard problems of the sort that certain morphological or [grammatical] agreement frameworks may provide ... it *does* appear to solve rapidly the modular problems that arise in natural morphological systems—at least for the examples described in chapter 6. In this respect, it corresponds more closely to natural grammatical systems than to morphological frameworks that are NP-hard or worse. We speculate that the separable character of natural morphological systems may yield this result. For example, one current line of phonological research known as *autosegmental theory* favors highly separable feature analyses; given our speculations, such grammatical systems may be more amenable to nonbacktracking and fast parallel algorithms than the two-level morphological system described in chapter 5 is.

To summarize, graph separability may provide a formalization for the notion of problem modularity that leads directly to efficient parallel algorithms. If this speculation is on the right track, then modular grammatical representations—whose components are separable in this way—may have precisely the right sort of structural characteristics for efficient parallel processing. If correct, such a view would direct us away from monostratal grammatical systems that attempt to characterize sentences at a single level of constraint and toward separable constraint systems such as ID/LP grammars planar separable constraint modules of government-binding theory, or autosegmental theory in phonology. However,..., not every kind of modularity supports modular processing; thus it is not an immediate result that separable constraint systems of this sort have significant advantages for parallel processing.

The caveat at the end is very important. If systems are highly intertwined in a deductive way—the kind of property usually highly prized in a principle-based theory—then planar separability may not ensue, and speedup may be impossible. Thus, the richly deductive structures *within* syntactic levels may not be susceptible to superfast parallel speedup. Even so, nobody has really investigated this possibility; it may be that there are still relatively autonomous subparts of principle-based theories that can be attacked by divide-and-conquer methods, or else subparts like autosegmental theory that use autonomous representational levels and so are efficiently analyzable.

On the other hand, there certainly are well-known autonomous representational levels in the usual T-model, and even some arguments (see Hornstein 1986) for autonomous subdivisions within the usual T-diagram at the level of logical form. On this account, further subdivisions of Move-α into distinct, ordered subcomponents would be computationally favored.

Considering parsing models, the two-stage division into structural assembly of phrases followed by binding,[69] This sort of autonomous modularity is highly separable, hence amenable to superfast parallel speedup of the sort that Kasif, Reif, and Sherlekar have explored. It remains to be seen just how the circuit models they propose can be turned into workable parallel implementations for grammatical theories, of course, but on balance, it appears that the modularity now appearing in most modern grammatical theories—from GB to GPSG—would have much to recommend it.

One must be cautious, however: so far as we know, just because a theory *is* highly interdependent does not mean that it is not amenable to parallel speedup, though this is an interesting possibility briefly discussed in Barton, Berwick, and Ristad (1987).

Whatever the outcome of these speculations, it is clear that our study of principle-based parsing has only just begun. Many parsing models are possible if one is willing to give up context-free rules; we only partially understand the relationship between modularity and computational complexity. Many topics in principle-based parsing—to name one, the relationship between skeletal $\overline{\text{X}}$ parsing, intonational contours, and precedence parsing—were not even addressed. But the moral should be plain: principle-based parsing *works*—witness the implemented parsers that use it. Principle-based parsers open up a whole new range of possible parsing algorithms—including those that don't even build phrase structure. As we better understand the representations and principles that underly grammars, it seems more and more certain that we'll see more parsers like these, providing a wealth of new parsing ideas and psycholinguistic predictions for many years to come.

[69]Though binding may proceed S by S, that is, in a pipelined way, before the entire sentence is processed) as proposed in Berwick and Weinberg (1984) and in part confirmed by the Freedman and Forster (1985) work. Crain and Fodor (1985:126) note that such modularity claims can indeed be controversial, since, as we have seen, modularity permits a greater variety of possible arrangements of logical information structures, hence a greater flexibility in adapting to almost any sort of experimental result. As they remark, "Frazier et al. have suggested that the constraints are in a different module than the rules, and are applied after the rules. But the constraints might be in a different module and yet applied simultaneously with the rules. And if they were, then the relative timing of processing operations would be indistinguishable from their relative timing in a nonmodular system."

Acknowledgments

Many people have contributed, directly and indirectly, to the ideas in this paper. I must first thank the natural language group at the MIT Artificial Intelligence Laboratory, who have provided a stimulating research environment in which to pursue the implementation of principle-based parsers: Ed Barton, Michael Brent, Bonnie Dorr, Sandiway Fong, Michael Kashket, and Eric Ristad. The particular contributions of Barton, Kashket and Dorr have already been noted. Sandiway Fong and I have discussed the virtues and vices of multidimensional representations, phrase markers, and phrase structure. Ed Barton deserves special credit: he has worked with me for several years on underspecified $\overline{\text{X}}$ parsing theory and has contributed many other stimulating ideas, as mentioned in the text.

I would also like to thank Stu Shieber, whose stimulating comments on this paper led me to think a bit more about logic grammars and compilation, as well as the other participants at the CSLI conference on natural language processing, in particular Janet Fodor, Bill Rounds, and Alexis Manaster-Ramer. Special credit goes to Tom Wasow and Peter Sells for arranging the conference, and to the Systems Development Foundation for its support of such efforts.

Steve Abney's research and Frazier's recent paper on principle-based parsing have also proved to be enormously fruitful springboards for thinking about these issues.

Noam Chomsky provided helpful commentary on an earlier draft of this material.

This work describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency under Office of Naval Research contracts N00014-80-C-0505 and N00014-85-K-1024. Further support was provided by National Science Foundation Grant No. DCR-8552543, under a Presidential Young Investigator Award.

References

Abney, S., 1986. A new model of natural language parsing. unpublished m.s., MIT Department of Linguistics and Philosophy.

Aho, A., and Ullman, J., 1972. *The Theory of Parsing, Translation, and Compiling*, Vol. 1. New York: Prentice-Hall.

Aoun, J., 1979. Indexing and constituency. Unpublished m.s., MIT Department of Linguistics and Philosophy.

Bach, E., 1965. On some recurrent types of transformations. In C. W. Kreidler (ed.), *Sixteenth Annual Roundtable Meeting on Linguistics and Language Studies.* Washington, D.C.: Georgetown University Monograph Series on Language and Linguistics, 18.

Barton, E., and Berwick, R. 1985. Parsing with assertion sets and information monotonicity. Proc. IJCAI-85, 769–771.

Barton, E., Berwick, R., and Ristad, E., 1987. *Computational Complexity and Natural Language.* Cambridge, MA: MIT Press.

Berwick, R., 1982. Locality principles and the acquisition of syntactic knowledge. Ph.D. dissertation, MIT Department of Linguistics and Philosophy.

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

Berwick, R., and Weinberg, A., 1985. Models for deterministic parsing. *Proc. NELS.*

Bowen, K. and Kowalski, R., 1982. Amalgamating language and metalanguage in logic. In Clark and Tarnlund, (eds.), *Logic Programming.* New York: Academic Press.

Chomsky, N., and Miller, G., 1963. Finitary models of language users. In Luce, Bush, and Galanter (eds.), *Handbook of Mathematical Psychology.*

Chomsky, N., 1981. *Lectures on Government and Binding.* Dordrecht: Foris Publications.

Correa, N., 1987. An attribute grammar implementation of Government-Binding theory. *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics.* Stanford, CA.

Crain, S., and Fodor, J.D., 1985. Rules and constraints in sentence processing. *NELS* 15.

Crain, S., and Fodor, J.D., 1985. How can grammars help parsers? in Dowty, Kartunnen, and Zwicky (eds.) *Natural Language Parsing.* Cambridge, England: Cambridge University Press.

Dennett, D., 1983. On the notion of rule-governed behavior. Proceedings of the Oxford Philosophical Society.

Dorr, B., 1986. Unitran: a principle-based translation system. S.M. proposal, MIT Department of Electrical Engineering and Computer Science.

Emonds, J., 1986. *A Unified Theory of Syntactic Categories.* Dordrecht: Foris Publications.

Fodor, J.A., Bever, T., and Garrett, M., 1974. *The Psychology of Language Uuse.* New York: McGraw-Hill.

Frazier, L., 1986. Natural classes in language processing. unpublished m.s., presented at the MIT Cognitive Science Seminar, November, 1986.

Frazier, L., Clifton, C., and Randall, J., 1983. Filling gaps: decision principles and structure in sentence comprehension. *Cognition* 13:187-222.

Freedman, S. and Forster, K., 1985. The psychological status of overgenerated sentences. *Cognition* 19:101–132.

Grishman, R., 1973. Implementation of the string parser of English. In Rustin (ed.), *Natural Language Processing*. New York: John Wiley.

Halle, M., 1986. Some speculations about the representation of words in memory. in V. Fromkin (ed) *Phonetic Linguistics: Essays in Honor of Peter Ladefoged*, New York: Academic Press, pp. 101–114.

Halle, M., and Vergnaud, J-R., 1987 forthcoming. *Stress and the Cycle.*

Hornstein, 1986. Semantics and quantifiers in a T-model. m.s., University of Maryland, College Park, MD.

Jaeggli, O., 1986. Passive. *Linguistic Inquiry* 17:4, 587–622.

Johnson, M., 1987 forthcoming. Parsing as deduction. Ph.D. dissertation, Stanford University.

Kaplan, 1973. A general syntactic processor. In Rustin (ed.) *Natural Language Processing.* New York: Algorithmics Press, 191–241.

Kasif, S., Reif, J., and Sherlekar, D., 1986. Formula dissection: a divide and conquer algorithm for satisfiability. m.s., Department of Electrical Engineering and Computer Science, The Johns Hopkins University, Baltimore, MD.

Kashket, M., 1986. Parsing a free-word order language: Warlpiri. *Proc. ACL*, 60–66.

Kartunnen, L. and Kay, M., 1985. In Dowty, Kartutunen, and Zwicky (eds.) *Natural Language Parsing*, Cambridge, England: Cambridge University Press, 279–306.

Klavans, J. 1982. Configuration in non-configurational languagse. *Proc. 1st West Coast Conference on Formal Linguistics.* Stanford, CA.

Kwasny, S. and Sondheimer, N., 1979. Ungrammaticality and extragrammaticality in natural language understanding systems. *Proc. 17th Annual Meeting of the ACL.*

Lasnik, H. and Kupin, J. 1977. A restrictive theory of transformational grammar. *Theoretical Linguistics* 4:3.

Lasnik, H., and Saito, M., 1984. On the nature of proper government. *Linguistic Inquiry*, 15:3, 235–290.

Marcus, M., 1980. *A Theory of Syntactic Recognition for Natural Language.* Cambridge, MA: MIT Press.

Matthews, R. 1984. The Marcus parser and grammatical theory. unpublished m.s., Rutgers University.

Platek and Sgall, 1978. A scale of context-sensitive languages related to the context-free languages. *Information and Control* 38:1, 1–20.

Pinker, S., 1984. *Language Learnability and Language Development.* Cambridge, MA: Harvard University Press.

Pylyshyn, Z., 1984. *Computation and Cognition.* Cambridge, MA: MIT Press.

Ristad, E., 1986. Complexity of linguistic models. S.M. thesis, MIT Department of Electrical Engineering and Computer Science.

Robinson, J., 1980. Diagram: A grammar for dialogues. SRI Technical note 205, February, 1980.

Saito, M., 1985. Some asymmetries in Japanese and their theoretical implications. Ph.D. dissertation, MIT Department of Linguistics and Philosophy.

Sells, P., 1985. *Lectures on Contemporary Syntactic Theories* Stanford, CA: Center for the Study of Language and Information.

Shieber, S., 1985. Using restriction to extend parsing algorithms for complex feature-based formalisms. *Proc. ACL*, 145–152.

Sharp, R., 1985. A parser for GB theory implemented in PROLOG. S.M. dissertation, Vancouver, University of British Columbia.

Slocum, J., 1984. METAL: The LRC machine translation system. ISSCO Tutorial on machine translation, Lugano, Switzerland.

Tenney, C., 1986. A context-free rule system for parsing Japanese. Cambridge, MA: MIT-Japan Sicence and Technology Program.

Thiersch, C., 1986. A prolog GB parser for German (personal communication).

Ullman, S., 1979. *The Interpretation of Visual Motion.* Cambridge, MA: MIT Press.

vanRiemsdijk, H., and Williams, E., 1986. *Introduction to the Theory of Grammar.* Cambridge, MA: MIT Press.

Weischedel, R. and Black, J. 1980. Responding intelligently to unparsable inputs. *American Journal of Computational Linguistics.* 6:2, 97–109.

Werhli, E., 1984. A government-binding parser for French. University of Geneva, Institute for the study of semantics and cognition working papers, no. 48.

CS-TR Scanning Project
Document Control Form

Date : 10/5/95

Report # AI-TR-972

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other: ____________

Document Information

Number of pages: 114 (120-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:

☐ Typewriter	☐ Offset Press	☒ Laser Print
☐ InkJet Printer	☐ Unknown	☐ Other: ____________

Check each if included with document:

☒ DOD Form	☐ Funding Agent Form	☒ Cover Page
☐ Spine	☐ Printers Notes	☐ Photo negatives
☐ Other: ____________		

Page Data:

Blank Pages(by page number): ____________

Photographs/Tonal Material (by page number): ____________

Other (note description/page number):

Description : Page Number:

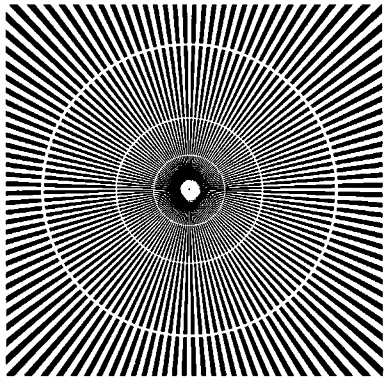
(A) IMAGE MAP! (1-114) UN#'ED TITLE & BLANK, 1-111, UN# BLANK,
(115-120) SCANCONTROL, COVER, DOD,
TRGT'S (3)

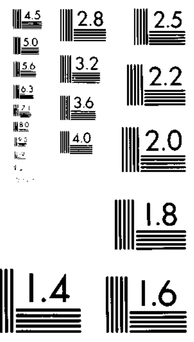
(B) CUT & PASTE FIG'S ON PAGES 35, 50, 68, 94, 96-97, 99, 102-103

Scanning Agent Signoff:
Date Received: 10/5/95 Date Scanned: 10/10/95 Date Returned: 10/12/95

Scanning Agent Signature: Michael W. Cook

Rev 9/94 DS/LCS Document Control Form cstrform.vsd



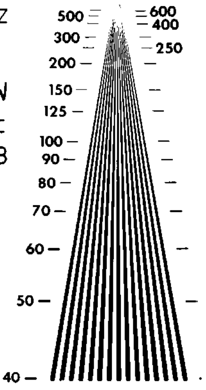


NMA MICROFONT QJKLPYZ
6BS12GH5D4X7U3W8V9E
PQR45DE9UV670FG8STHIJNOWXABYZ
3KLM12C IJK90FGH8STU67AB45CDEZ123VWNOPXYQRLM

ABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrst
uvwxyz0123456789 OCR-B

ABCDEFGHIJKLMNOPQRSTUV
WXYZabcdefghijklmnopqr
stuvwxyz1234567890PICA

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Elite



ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 6 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 4 pt

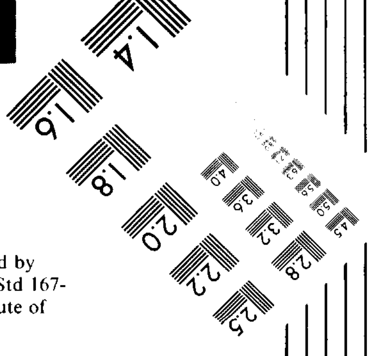
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 8 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 10 pt

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 Spartan Medium 12 pt

IEEE Std 167A-1987

FACSIMILE TEST CHART



AIIM SCANNER TEST CHART # 2

Spectra

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Times Roman

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Century Schoolbook Bold

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

News Gothic Bold Reversed

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Bodoni Italic

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:",./?$0123456789

Greek and Math Symbols

4 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋟⋞⋛⋜≡

6 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋟⋞⋛⋜≡

8 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋟⋞⋛⋜≡

10 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓",./≦±=≠°><⋟⋞⋛⋜≡

White

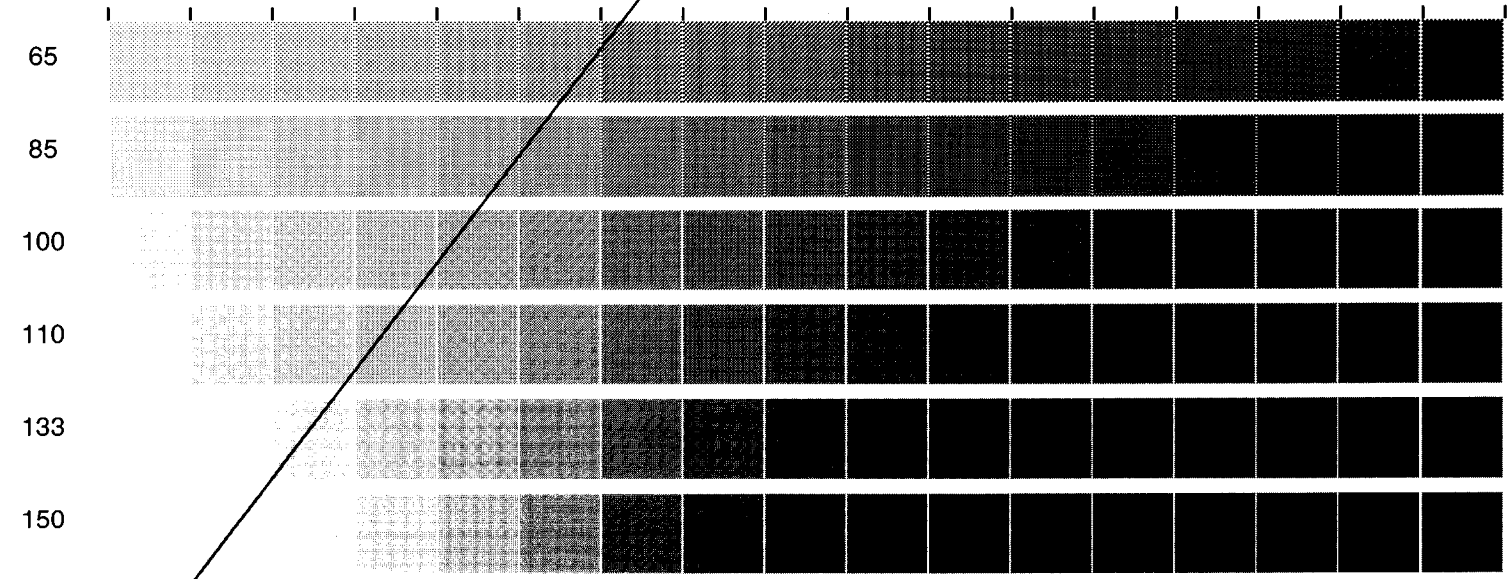
Black

Isolated Characters

e	m	1	2	3	a
4	5	6	7	o	°
8	9	0	h	l	B

A4 Page 6543210

MESH HALFTONE WEDGES



A4 Page 6543210