In Artificial Intelligence at MIT: Expanding Frontiers
Ed. P Winston MIT Press 1990

12

Open any textbook on Artificial Intelligence and you will find a chapter on language that shows how a few pattern-matching rules enable toy-world sentences to be transformed into syntax-reflecting parse trees. Thus encouraged, you might suppose that handling real-world sentences is just a matter of a few more rules.

Alas, just a few more rules are inadequate, and many research groups have demonstrated that many thousands are not enough, evoking doubts about whether traditional pattern-matching rules are really the right representation for expressing linguistic competence.

But there is an alternative approach. Instead of an unmanageably large, undifferentiated rule set, Berwick and Fong work with a dozen or so explicitly parameterized modules. Although each module has only a few parameters at most, they combine multiplicatively, explaining the rich variety found in the world's languages.

One of the many advantages of the approach is economy of implementation. You should be able to condition a parsing program to handle your favorite language by selecting the appropriate parameter settings and by supplying the appropriate lexicon. Berwick and Fong have one set of parameter settings that capture the conventions of English. Other sets deal with Spanish, German, and even Warlpiri, an exotic, much studied language, cherished by linguists for its strangeness, spoken only by a few thousand central Australian aborigines.

Principle-Based Parsing: Natural Language Processing for the 1990s

Robert C. Berwick
Sandiway Fong

Introduction: Principle-Based Parsing and Rule-Based Parsing

This chapter describes a new approach to processing natural language, *principle-based parsing*, that has been developed at the MIT Artificial Intelligence Laboratory over the past five years. Principle-based parsing replaces the traditionally large set of rules used to parse sentences on a language by language basis with a much smaller, fixed set of parameterized, universal principles. The principles interact deductively to replace many rules.

We have used this approach to implement a unified parsing scheme that can solve many thorny problems in natural language processing:

- Handle a wide variety of languages, including English, Spanish, German, and even "exotic" language like the Australian aborigine language Warlpiri where words can occur in virtually any order.
- Translate single sentences from one language to another.
- Optimize natural language parsing for use on serial or parallel computers.

This chapter reviews each of these accomplishments.

What is principle-based parsing? Perhaps it is easiest to say what it is *not*. It is not like more traditional parsing that relies on many thousands of individual, language-particular rules, exemplified by augmented transition network systems (ATNs) or most systems based on context-free grammars.

These rule-based parsing systems attempt to describe sentences typologically by spelling out shallow word surface order and spelling patterns—passive, dative, and the like.

For instance, consider a "passive" sentence: *The ice-cream was eaten.* To understand this sentence, at the very least a parser must be able to analyze *ice-cream* as the object of *eat*—the thing eaten. This is triggered (in English) by the form of the verb *be* (here, *was*) plus the *en* ending on the verb *eat*. A typical rule-based parser might capture this left-to-right pattern in an IF-THEN rule:

IF:	subject filled *be*–Verb+ed No object
THEN:	make the subject the object

Note that this IF-THEN rule directly encodes the left-to-right order of the English passive pattern, along with all its particular features. It is appropriate only for English, and only for this particular kind of passive form.

The basic idea of principle-based parsing is to replace this shallow rule with a much deeper, smaller, explanatory set of basic principles. In some ways, this is much like the shift in medical expert systems from a shallow, descriptivist approach to an explanatory theory based on, for example, a knowledge of kidney physiology.

The motivation for the shift from rules to principles is the same in both domains. Rule systems have many problems. They are too inflexible, too specific, too fragile, too hard to maintain, and too large. As we will see, principle-based parsing repairs each of these defects:

- What happens when a sentence is only partially well-formed? Sentences like *What do you wonder who likes*, or *John is proud Bill*, though hard to understand, do not cause people to collapse like a rule-based system. Rather, people understand such sentences uniformly. Rule-based language systems have traditionally handled such possibilities by adding weights or more rules that can describe the wrong sentences. But this makes the rule set larger still.
- What happens if the rule system is missing a rule that is almost, but not quite, like the one that handles passives? Consider the sentence *The ice-cream got eaten.* This is a simple dialect variant, but unless it has been preprogrammed into the rule base—often one programmer's dialect—the full system will fail on such an example. Of course, once the problem is known a system can be patched by adding a new rule, but there is no end to the patches, the maintenance problems, and the size of the rule system.
- What happens with other languages? The French sentence *faire manger la pomme par Jean* ('was eaten the apple by John') is like the English passive, but there is no *be* form, and the object can follow *eat*

(*manger*). Thus the entire system must be re-tooled for new dialects and languages, and this in fact has been the traditional approach. In a principle-based approach, we can view such cases as variations on a basic theme. Instead of writing a completely new, specialized IF-THEN rule, we can regard the French (and English) examples as parameterized variants of a set of more primitive, underlying components. The section "Principle-Based Translation," in this chapter and the chapter by Dorr in this volume describe principle-based translation in more detail.

- What happens when the rule set becomes too large? Because rules are fine-grained and language particular, existing rule-based natural language systems use thousands of rules. Because parsing algorithms run as a function of grammar size, an inflated rule size forces poor system performance. Too much effort is expended trying to build special-purpose algorithms or hardware when the real source of the problem is an overly-large rule system. For example, one recent language system developed at Boeing contained many thousands of individual rules just for a portion of English. As a result, even a single sentence could be analyzed in a thousand different ways or more.

A few principles can replace many rules

In contrast, a principle-based approach aims to reconstruct the vocabulary of grammatical theory in such a way that surface and language-particular constructions like passive *follow* from the interactions of a small set of primitive elements. Figure 1 illustrates the difference. The top half of the figure shows a conventional rule-based approach. Each sentence type (a *construction* like passive) is described by a different rule. The bottom half of the figure shows a principle-based approach.

Principles and word meaning building blocks are like atoms in chemistry, or axioms in a logic system. By combining just a few dozen atoms, we can build up a huge number of chemical compounds (sentence rules and word meanings) instead of listing each compound separately. In the language domain, we can replace the surface effect of many rules with longer deductive chains of just a few axioms. Note that one can get the multiplicative effect of $n_1 \times n_2 \times \ldots$ rules by the interaction of $n_1 + n_2 + \ldots$ autonomous modules. Thus, by supplying a dozen or so principles, each with two or three degrees of freedom, we can encode many thousands of rules. By varying the parameters, we can describe different dialects and even different languages. Naturally, no single principle accounts for all the variation we see in a language, just as no single molecule accounts for all chemical compounds and reactions. It is the interaction that matters.

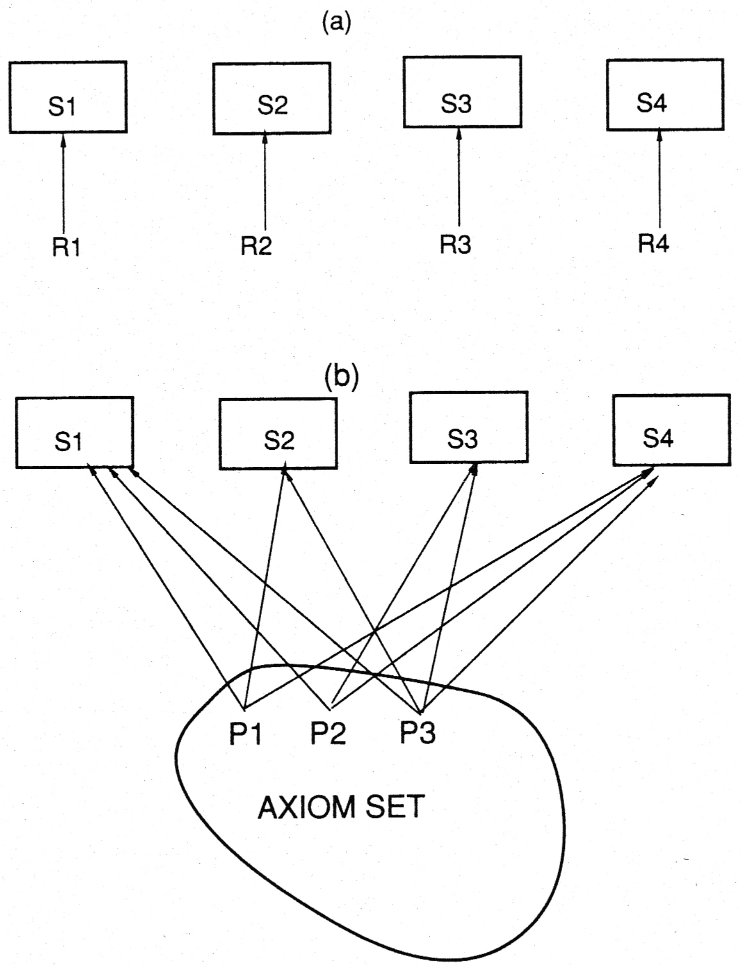


Figure 1. These two figures illustrate the difference between rule-based and principle-based systems. (a) A rule-based system. Each sentence construction type, like passive, is described by a distinct rule. (b) A principle-based system. Sentence types are derived from a smaller basis set of more fundamental principles that deductively interact to yield the effect of constructions like passive. See the section on "Principles and Parameters in Parsing Design" for more details on just how this works.

An outline of things to come

Of course, the principle-based approach raises many questions for parsing. The next three sections of this chapter will answer each of these three key questions in turn:

1. Is principle-based language analysis possible? Can one describe all languages, or even one language, as the interaction of a small, universal set of principles?
2. Can one build parsers that use principles instead of rules?
3. Can one make these parsers computationally efficient?

While we will defer detailed answers to these questions we can offer a glimpse at what's to come now.

Consider first a simple example of how general principles can replace specific rules. One general principle says that action-expressing phrases in a sentence must either *begin* with a verb in some languages, or *end* with a verb in others. This basic description of the tree shapes in a language, dubbed $\overline{\text{X}}$ *theory*, gives us part of the variation between languages like English and Spanish on the one hand, and languages like German and Japanese on the other. In English, the verb must come first, with the object after. A second principle, called the *Case Filter*, says that all pronounced nouns like *ice-cream* must receive case, either from an active verb like *ate* or an auxiliary verb like *was*; the adjective-like verb *eaten* does not do the job. Taken together, these two principles plus a few others conspire to push *ice-cream* into its position at the front of a sentence if there is a verb sequence like *was eaten*. There is no explicit passive rule.

The section on principle-based parsing for Warlpiri shows how these principles can be used to build a parser that does not use individual rules. Drawing on research carried out by Michael Kashket [1986], it also shows how the same principles can be made to work with a language that is very different from Romance languages or German, in this case, the Australian aborigine language Warlpiri. Warlpiri is a good test case for the principle-based model because its structure seems at first so different from that of English, German, or Romance languages. This section also briefly describes a related design for a principle-based parser, used by Dorr [this volume] and described in more detail in the section on principle-based translation.

Many principles operate like constraint filters. This lets them handle the problem of partially well-formed sentences. A principle-based parser can accommodate language "mistakes" by constraint relaxation. If a sentence is ill-formed, it is simply because one or another principle fails to hold. In fact, in the principle-based approach, there really is no such thing as an ungrammatical sentence—this notion doesn't even really apply. Every string of sounds is assigned some interpretation. Some of these happen to "pass" all the principles, while some fall short in one area or another.

For example, consider the sentence, *This is the ice-cream that I don't know whether it was eaten by John.* Technically this sentence is ill-formed, and it would break existing language interfaces because they would have no special rule that could apply in such cases. In contrast, a principle-based system would degrade gracefully. In this example, a locality principle that limits the distance between words like *it* and *ice-cream* is at fault, but importantly all other principles hold. In particular, the principle that every sentence has a subject (*it*) and a verb (*eaten*) still holds. Sentence analysis would proceed as before, simply taking note of this violation, which does not impede inference or understanding. No special weights or extra rules are required.

The principle that a sentence needs a subject and object is also parameterized, illustrating how principles can ease the task of language translation. In languages like Spanish and Italian, and under degraded situations in English, the subject need not be expressed. These and other similar, but simple, parameters are what makes English different from Spanish or Italian. Thus our parser does not need special rules to analyze Spanish or another English dialect; it simply sets its parameters to those values required for Spanish, such as "the subject may be missing." Only the dictionary changes from language to language. The section on principle-based translation in this chapter and Dorr's chapter in this volume describe an implemented system, UNITRAN, that adopts this approach.

The final section of this chapter turns to the computational optimization of principle-based parsers, carried out by Sandiway Fong at MIT. Because a principle-based language processor uses several independent constraint modules, it also becomes possible to optimize its performance and test it under a variety of processing assumptions. One can develop an automatic "compiler" (really a source-to-source translation procedure) for such constraint systems, because we expect different dialects and different languages will demand different principle processing topologies for optimal performance. For instance, in a language like English, information often comes at the very beginning of phrases to tell us what the phrase will be like: *the* tells us that we will be describing an object next. But in German or Japanese this information may be delayed until the end of a sentence. Thus different languages, and even different sentences within the same language, might optimally use different processing strategies. Instead of hand-coding these, we would like to automatically, perhaps dynamically, guide the application of constraint modules.

Two general ideas guide principle-based parser design. First, certain modules are logically dependent on others. For example, the locality principle that calculates whether *it* is too far away from *the ice-cream* must use phrases, but phrases are fixed in part by the principle that says whether a verb comes first or last in a phrase. This dependency structure provides a kind of flowchart that is amenable to conventional computer science techniques like dataflow analysis.

Second, as a general condition one should apply the strongest constraints as early as possible while delaying hypothesis-space expanding procedures as long as possible. Well-established techniques exist to impose this ordering among modules by estimating the filtering/expansion power of different principles. Fong calls his resulting design a *principle-ordering parser*. He shows that order-of-magnitude improvement is sometimes possible by optimal ordering.

Principles and Parameters in Parsing Design

To begin, let us answer the first question about principle-based parsing raised earlier: can a small number of principles describe natural language? This is a question about linguistic description, so we rely on linguistic theory to answer it. Currently, we have adopted a variant of the *principles-and-parameters* theory developed at MIT and elsewhere (sometimes called *government-binding* or *GB theory* [Chomsky 1981; Lasnik & Uriagereka, 1988]). Figure 2 shows the topology of the system currently used. It pictures about a dozen modules or *theories*, most of which are described more fully in in the next paragraph. Lines between the modules mark logical dependencies—certain constraints are defined in terms of others.

Let's see just what these principles mean and how they work together to account for passive sentences.

- $\overline{\text{X}}$ **theory** describes the basic tree shapes allowed in a language. Roughly, natural languages allow two basic tree forms: function–argument form, as in English where the verb begins a verb phrase, a preposition begins a prepositional phrase, and so forth (*eat the ice-cream, with a spoon*); and argument–function form, as in Japanese and much of German.
- The **Theta Criterion** says that every verb must discharge its thematic arguments—its placeholders that flesh out who did what to whom. Thus a main sentence with *eat* must mention the eater and optionally the thing eaten, whereas *put* must mention the thing that is put somewhere and the location it is put (one can't have *John put the book*).
- The **Case Filter** says that pronounced (*overt*) noun phrases like *ice-cream* must receive Case. What is meant by case? In simplest terms, it is much like what is found in a traditional Latin grammar: the subject noun phrase receives nominative case; the direct object of the verb receives accusative case; the object of a preposition receives oblique case. The pale residue of this *case-marking* system shows up in English in the use of *her* as an object versus *she* as a subject: *Mary likes her*; *She likes Mary*. This is what accounts for the difference between sentences like *It is likely that John will win* versus the (ill-formed) *It is likely John to win*. In the first sentence, *John* receives nominative case from *will*; in the second, there is no tensed verb or verb-like element to give *John* case, and so the sentence violates the case-filter.
- **Binding Theory** spells out how pronouns may be related to their antecedents in certain configurations. For example, compare the sentences,

  John thinks that he likes ice-cream
  He thinks that John likes ice-cream

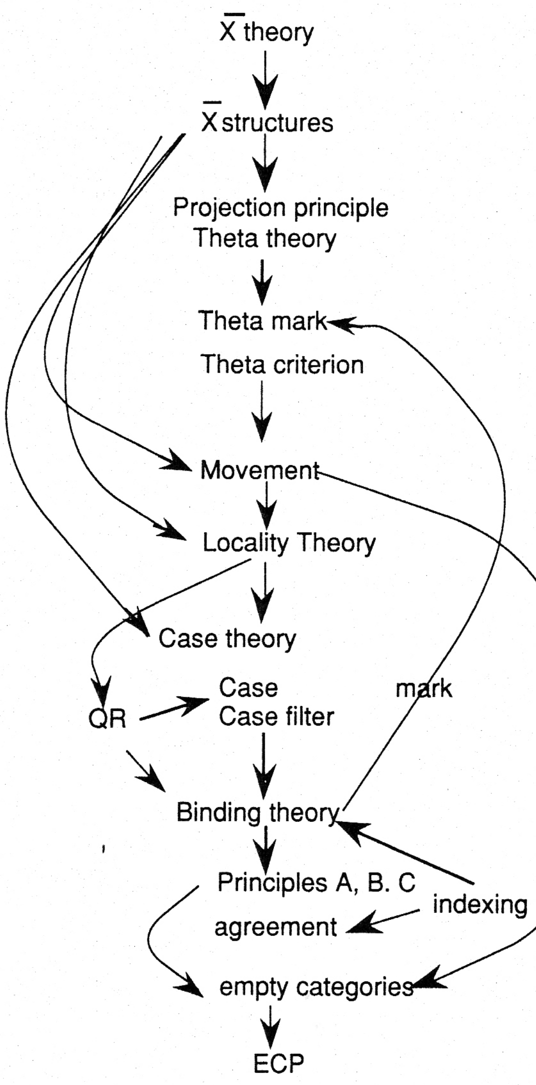


Figure 2. A design for a principle-based parsing system. Each module component varies over a limited parametric range. The joint interaction of all modules replaces the effects of many particular language rules.

In the first sentence, *John* and *he* may refer to the same person, while in the second, they cannot.

- **Locality Theory** and the **Empty Category Principle** restrict where "silent" noun phrases (*empty categories*) can appear. A silent noun phrase is not pronounced, but still needed to understand a sentence. For example, in the sentence below,

  John wants to like ice-cream

there is a silent noun phrase, that we can denote *e*, that acts as the subject of *to like ice-cream*; just like a pronoun, it refers to *John*:

John wants to e like ice-cream

Empty categories cannot appear too far away from their antecedents (locality) and only in certain configurations (the empty category principle or *ECP*). The first example below shows a violation of locality—the empty category *e* is too far away from *John*—and the second a violation of the ECP. (We will not be concerned in this chapter with exactly how constraints like the ECP are formulated.)

John seems it is certain e to like ice-cream
John was wanted to e like ice-cream

- The **Movement Principle** says basically that any phrase can be moved anywhere. For example, we can change *John likes ice-cream* to *Ice-cream, John likes*. (Of course, this freedom may violate other principles.)

Having covered these basic principles, we can now see in detail how they interact to yield *the ice-cream was eaten*. If we think of the principles as axioms, the passive construction emerges as a theorem. But the deductive chain is much longer than in a simple IF-THEN rule system, where there is a direct, one-step connection between passive sentences and rules. The following sequence outlines the steps:

$\overline{\text{X}}$ theory sets the basic function-argument order of English

↓

was eaten the ice-cream

↓

Eaten is an adjective, and so does not assign case

↓

Ice-cream must receive case

↓

So *ice-cream* moves to subject position
where it receives nominative case

↓

This leaves behind an empty category, linked to *ice-cream*
(so that *eat* can meet the Theta Criterion and
make *ice-cream* the thing eaten)

↓

the ice-cream was eaten e

This may seem like a lot of deductive work for one sentence, but the important point is that the *same* principles combined in different ways yield different sentences, just as the same molecules can combine in different ways to make many different chemical compounds. For example, in the sentence,

It was believed that the ice-cream was eaten

no movement is required because *ice-cream* already receives nominative case from *that*. (This would show up more clearly if *ice-cream* were replaced with a pronoun. Then the pronoun would have its nominative form: *It was believed that he was eaten.*)

Principles and parsing

So far, all that we have done with principles is describe sentences. How can we use principles to parse sentences—that is, to assign structure to sentences that shows what the subject and object are, what the thematic roles are (who did what to whom), and so forth? In some way we must reproduce the deductions that connect axioms to sentences. Of course, all we have to start with is the input sentence, a dictionary, and the principles themselves.

While there are several possible approaches, it is useful to divide the principles into one of two classes: *generators* and *filters*.

Generators produce or hypothesize possible structures. For example, consider $\overline{\text{X}}$ Theory. Given a string of words, say, *eat the ice-cream*, $\overline{\text{X}}$ theory would say that *eat* is possibly the beginning of a verb phrase, with *the ice-cream* as its argument. Similarly, Movement Theory creates possible structures. Given a valid $\overline{\text{X}}$ structure, Movement Theory can displace various noun phrases like *ice-cream* to create new ones. Binding Theory also generates new output possibilities from old ones. For example, given the sentence *John thinks that he likes ice-cream*, *he* can refer either to *John* or someone else not mentioned in the sentence, thus generating two candidate outputs.

Filters weed out possible structures. Most of the remaining boxes in the module picture are filters—the Case Filter, the Theta Criterion, the Empty Category Principle, and Locality Theory. For example, if the structure *John is proud ice-cream* is input to the Case Filter, it would be filtered out as a violation (it should be *proud of ice-cream*, where *of* assigns case to *ice-cream*).

Given this generator-filter model, the simplest way to build a parser is as a cascaded sequence of principle modules, as shown in figure 3(a). (For reasons of space the figure does not show all the possible principle modules.) The input sentence, *the ice-cream was eaten*, passes into the first module, in the figure $\overline{\text{X}}$ Theory, which produces several output possibilities indicated by multiple arrows (depending on word and structural ambiguities). The basic point is that these hypotheses are driven from the input in a bottom-up way. Given a verb, the system posits that a verb phrase must start; if it is a preposition, then a prepositional phrase must begin, and so on. Note that this requires access to a dictionary. As usual, these hypotheses are subject to numerous ambiguities (words may be either nouns or verbs, for example), but we will defer these standard problems here. All the usual techniques for efficient bottom-up processing, such as lookahead, can be useful here.

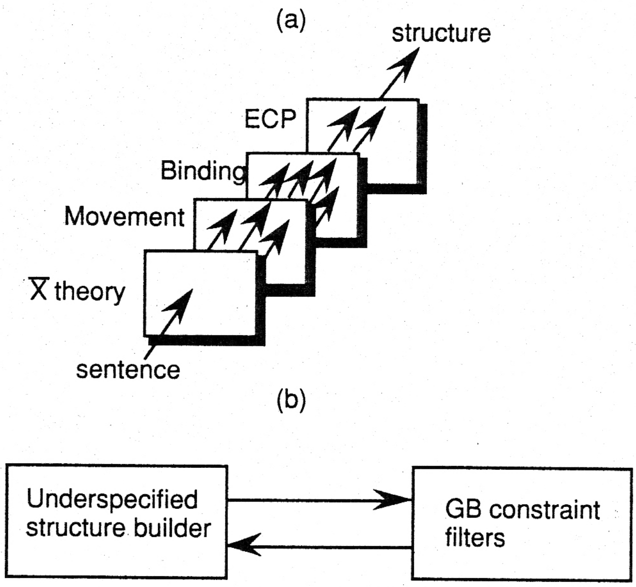


Figure 3. Principles may be classified as *generators* and *filters* and then organized into parsers in a variety of ways. (a) A sequential parser design. The input sentence is successively expanded and collapsed into a series of structural hypotheses. Generators expand the hypothesis space, while filters narrow it down. (b) A coroutine parser design, as used by Dorr in her translation system. Structure-building generators operate in tandem with constraint modules.

Alternatively, such a system could enumerate all possible function-argument structures *before* even looking at the input until it hits upon one that matches the input. Such a straightforwardly hallucinatory approach is known to be fraught with hazards unless special precautions are taken (it may not terminate, for example). For this reason, almost all existing principle-based systems attempt to access the information in the input sentence as quickly as possible, and this will be the case in all the principle-based parsing systems described in the rest of this chapter.

Continuing with our rough conceptual picture, the hypotheses output from the $\overline{\mathrm{X}}$ component are fed into the next module down the line, the Movement component, which also expands the number of hypothesized structures. Binding also generates multiple hypotheses. Finally, the ECP whittles down these multiple hypotheses to just one: the output structure *the ice-cream*$_i$ *was eaten* e_i (the subscript i indicates that the empty category following *eaten* is linked to *ice-cream*). In each step, important information from a dictionary or lexicon may be accessed. For instance, it is the lexicon that says just what thematic roles a verb requires—that *eat* needs an eater and an optional thing eaten. This information must figure in the hypotheses that are generated and filtered.

Because many of the constraints depend on particular structural configurations as inputs the principle modules can only be ordered in certain ways. For instance, case is often assigned only under a particular local structural arrangement—the element receiving case is an immediately adjacent sister to a verb or a preposition. These logical dependencies must be respected in any principle parser design.

As an alternative to the straightforward sequential design it is possible to *coroutine* the operation of filters and generators, alternating between structure-building $\overline{X}$ and movement components, and filters (part (b) of figure 3). This approach has been adopted by Dorr for her translation system (see the "Principle-Based Translation" section). Dorr exploits the structure information in $\overline{X}$ theory to drive an Earley-type parser that is coroutined (operates in tandem) with filtering constraints. Whenever the $\overline{X}$ parse can no longer be extended, filters and some generators are called to weed out underconstrained parsers or propose parsing extensions. See the next section for more details on this coroutine design, and the final section for a more systematic framework in which to evaluate these different architectures.

Having seen how a principle-based parser might work in broad outline, we now review two important principle-based parsers developed over the past three years at the MIT Artificial Intelligence Laboratory: Kashket's Warlpiri parser and Dorr's translation system. The Warlpiri parser will be described first, in the rest of this section while Dorr's parser will be covered in the "Principle-Based Translation" section.

Principle-based parsing for Warlpiri

Warlpiri provides a good testbed for the principle-based parsing approach because it seems on the surface to look very different from English, German, or the Romance languages. Warlpiri word order is quite free. Even so, Kashket [1986] shows that the difference between a parser for Warlpiri and one for English is roughly a parametric difference in case marking: when the verb marks case, as in English, then this tends to fix word order; while if other elements mark case, as in Latin or Warlpiri, or some parts of English, then word order tends to be free. Let's examine how Kashket's model works.

By using a vocabulary other than the *concatenation* and *hierarchy* that are blended in context-free rules we can easily account for the free-word order found in Warlpiri *as well as* that part of English that appears to exhibit fixed-word order (subject-verb-object) and those parts of English that are relatively free (prepositional phrases). One and the same parser will work for both. In contrast, because context-free rules can use 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.

Warlpiri and rule-based parsing

To begin, let's consider some variations in a simple Warlpiri sentence. All permutations are legal. (Hyphens are added for readability):

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

'I am taking the boomerang from the child'

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

'From the child I am taking the boomerang'

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

'It is the boomerang I am taking from the child'

(Plus 21 other possibilities)

Although phrase order is free except for the rigid auxiliary verb-like element 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 in argument-function form, or what is called 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 traditional 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, while NP stands for a noun phrase.)

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 though, this approach ignores the basic and well-known asymmetry between subjects and grammatical functions like direct objects and indirect objects (see Laughren [1987] for discussion). For instance, it is the subject, not the object, that can be empty in constructions such as *I wanted to leave*, which does not have a counterpart *I wanted Bill to leave* meaning I wanted Bill to leave me. This asymmetry leads directly to positing a certain hierarchical structure that explicitly represents the domination of the object by the verb, with the noun phrase subject external to the verb phrase. Thus a better context-free grammar would be something like this:

S → NP-S VP
VP → V NP-O

But as is plain, this sort of grammar cannot parse the sentence order V NP-S NP-O that is observed in Warlpiri:

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

'taking am I the boomerang from the child'

To get over this hurdle various proposals have been made: invisible verb phrase nodes, movement rules, 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.

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 [Kartunnen & Kay 1985] is to say simply that one particular phrasal order is stored and the permutations that actually appear are generated on demand. The base grammar would remain small. As Kartunnen 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.

Parsing Warlpiri with principles

Kashket's key insight is to apply case marking and case assignment principles for Warlpiri at two autonomous representations. 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:

- *Precedence* structure: This level expresses, among other things, precedence relations (one morpheme precedes or follows another).
- *Syntactic* structure: 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 almost all principle-based predicates for syntax).

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/fixed word-order phenomena.

Case marking is taken to be an essentially word based (or phonological) process, hence one that logically ought to be represented at the level of precedence structure. Therefore, case marking depends upon precedence information, because this is encoded at the morphemic 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. Let us see how this works in an example.

In the word *ngajulu-rlu*, the ergative case marker *rlu* case marks *ngajulu* (English *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 4.

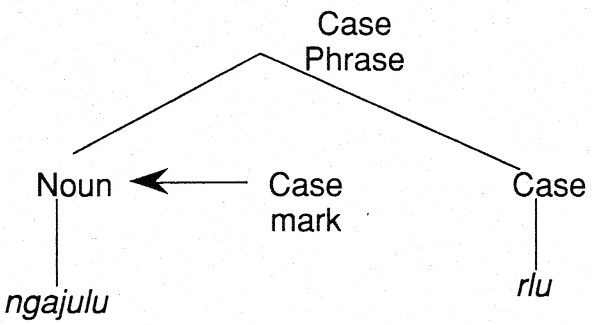


Figure 4. Directional case marking forces fixed internal word order in Warlpiri. The "heads" of Warlpiri words are final, with case marking to the left.

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 Warlpiri. 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.

Case *assignment* is carried out at the phrasal level under sisterhood, but with one crucial difference: because 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*).[1] 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 $\overline{\mathrm{C}}$ 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 $\overline{\mathrm{C}}$ phrase. Note that all three words are in argument-function (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 verb morpheme unit is now projected into syntax under the usual $\overline{\mathrm{X}}$ format: it contains a node, a $\overline{\mathrm{V}}$ node, and a $\overline{\overline{\mathrm{V}}}$ node. Under Kashket's model, the $\overline{\mathrm{V}}$ node assigns absolutive or dative case (in either direction);

[1] The AUX unit will be ignored for these and all remaining examples.

because *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	↔	ERGATIVE
THEME	↔	ABSOLUTIVE
PATH	↔	DATIVE

Figure 5 shows the resulting syntactic tree. It is important not to be misled by the order of the subtrees shown in this structure. While one must write the subtrees in some order, in fact the first $\overline{C}$ could have been written to the right of the $\overline{V}$ instead of on the left.

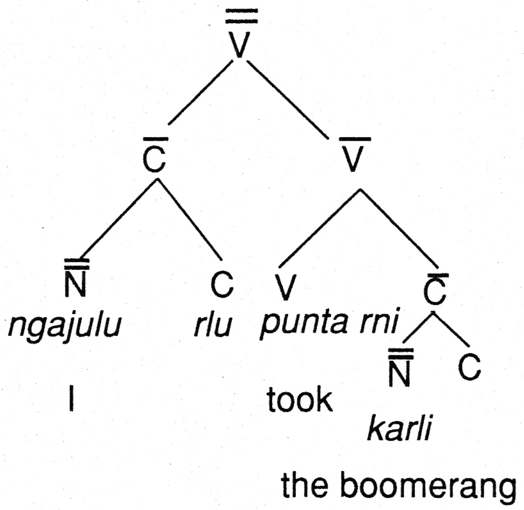


Figure 5. A Warlpiri syntactic structure for *I took the boomerang*. It is important to note that the order of subtrees is not encoded at this level, even though the picture must show some order on the page.

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 a verb *is* a case marker, so position matters: the verb case marks subject and object at the morpheme level where linear precedence is encoded. (More precisely, it is the verb's tense that marks the subject's nominative case to the left.) The result is that we usually get the order subject-verb-object in English.

Note that English also exhibits some phrase order freedom: prepositional phrases may appear relatively freely after a verb. Under the current account, that is because case assignment is carried out internally by the preposition. There is no ordering among phrases.

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: one for precedence structure and one for syntactic (hierarchical) structure. These two operate in tandem. Input sentences are passed to the precedence-based lexical parser, which breaks words into morphemes and outputs an ordered forest of trees. Every morpheme is also sent to the hierarchical parser, which projects information based on the lexicon and 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. The 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. Table 1 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 details are omitted here.)

The lexical parser determines the well-formedness of words according to morphological 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 verb node produced, as described earlier. A dictionary is consulted here, as is typical. In addition, if a verb projection (predicate) is being formed, the dictionary will supply case assignment actions to be associated with the projections of the verb, 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. 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 associate case assignment actions with 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.

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: -

Table 1. The Warlpiri dictionary directly encodes $\overline{\text{X}}$ features, case marking, and case assignment.

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.)

An example parse

An example parse should make the 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, precedence selection), forming a complete word with case as the root of the phrase. The next morphemes comprise the auxiliary and are projected as an auxiliary phrase by a procedure not described here. Third, the verb *take* with its tense *rni* marker is encountered. The tense selects the verb to its left, forming a verb 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 verb unit 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 6 gives a snapshot of the hierarchical structure built 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. Note that the tense element *rni* has been removed from the verb (it is attached to the auxiliary unit at this level of representation).

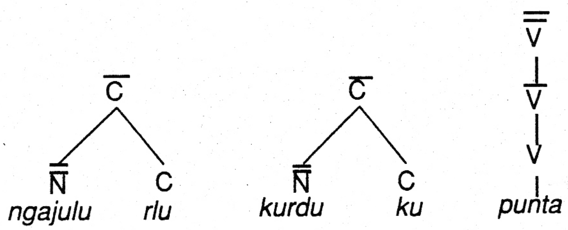


Figure 6. Starting to parse a Warlpiri sentence. The first three words, up through the verb and to the subject that follows it, have been analyzed. This figure represents hierarchical structure, so in fact the tense element *rni*, which is part of the verb at the precedence structure level, does not appear here.

Now any actions attached to the V(erb) projected nodes apply if possible. Because there is an ergative case-marked argument, ergative case assignment applies, linking $\overline{C}$ to the V phrase. Similarly dative assignment is possible and *kurdu-ku* is attached (see figure 7).

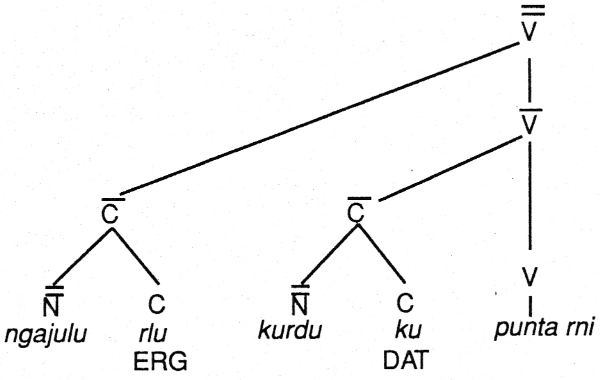


Figure 7. Linking the ergative subject and dative object in a Warlpiri sentence. These actions are forced by the dictionary entries associated with projected verb elements.

Because 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 $\overline{\mathrm{C}}$ to the phrasal parser. Once again, a V projection action can now apply to this case marked argument. This action assigns *karli* absolutive case (the THEME of the sentence). Figure 8 shows the final result. "PS" refers to the level of word structure, or precedence structure, while "SS" is hierarchical phrase structure. The figure also shows how the tense marker *rni* is attached as part of the auxiliary or inflection phrase that dominates the entire phrase structure.

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 to 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

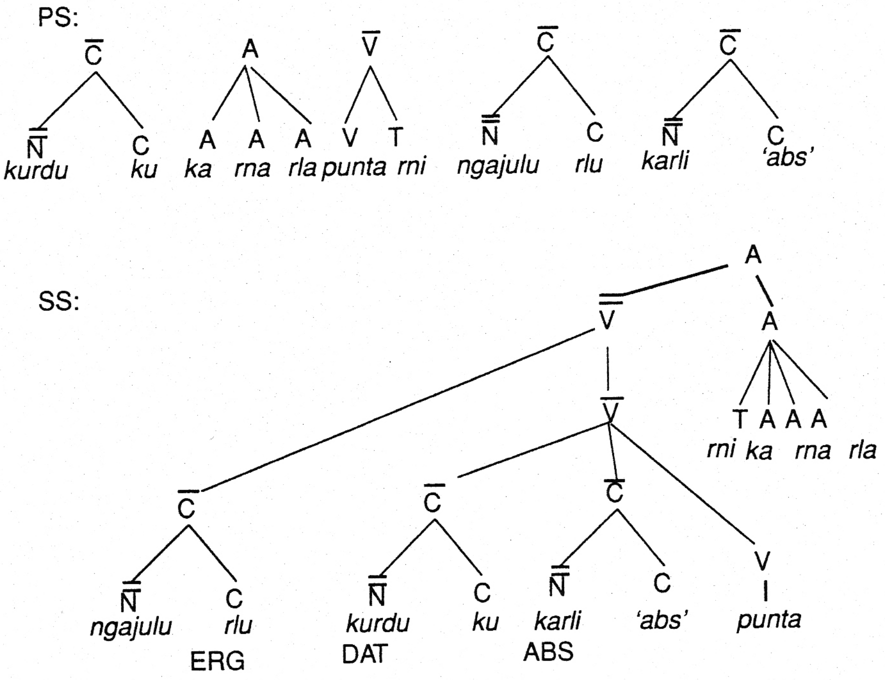


Figure 8. Linking in the Absolutive argument—which is phonologically absent—in the Warlpiri example sentence gives us the THEME of the sentence. This figure shows both the final precedence (PS) and syntactic structures (SS).

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 as a difference between which elements case mark and which do not, all without resorting to novel parsing procedures for either languages.

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 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.

Principle-Based Translation

We next turn to a principle-based parsing for a different domain: language translation. We describe a particular principle-based translation system, implemented by Dorr [1987]. Dorr's system successfully overcomes the difficulties of a rule-based approach. It also illustrates a coroutine design for principle-based parsing, that interleaves generators and filters. (Dorr's chapter in this volume shows how her system handles some of the other difficulties in translation.) We will see how this work on a Spanish example sentence that would ordinarily require a number of complex rules for its description:

¿Qué vio?

This short sentence is deceptively simple. It actually shows three interesting phenomena: a null (missing) subject; inversion of the verb; and movement of *qué* to the front of the sentence. A traditional rule-based system would describe each of these explicitly. As we shall see, Dorr's system can describe them all via the same parameters that are required for English.

Dorr's parser works by using a standard context-free parser, the Earley algorithm, on very slightly expanded $\bar{X}$ skeleton tree structures. These skeletons guide the Earley algorithm parser working in tandem with other principles. By modularizing the principles in this way significant computational efficiencies are realized.

Dorr assumes that $\bar{X}$ theory provides the basic phrase configuration possibilities (across all languages). The basic rules,

$$\bar{\bar{X}} \Rightarrow (\text{Specifier})\ \bar{X}$$
$$\bar{X} \Rightarrow X\ (\text{Complement})$$

are combined with two rules to handle adjuncts,

$$\bar{X} \Rightarrow (\text{Adjunct})\ \bar{X}\ (\text{Adjunct})$$
$$\bar{\bar{X}} \Rightarrow (\text{Adjunct})\bar{\bar{X}}\ (\text{Adjunct})$$

where parenthesized symbols are optional.

Some terminology is useful here. A *specifier* is simply a word like a determiner that further specifies the properties of a phrase. A *complement* is just what we have been calling the *arguments* of a verb or a preposition. An *adjunct* is an optional phrase that need not be part of a verb's thematic structure.

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 (assuming just one level of recursion) the $2^2 = 4$ possible tree topologies shown in figure 9, where (a) corresponds, for example, to English phrase structure order. What Dorr has done, then, is to partially "compile out" information about skeletal phrase structure possibilities. Note that these skeletons do provide topological information for parsing but do *not* provide any detail about categorial identity or verb selection information—for instance, whether *eat* must take an object or not. 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 parameters in each particular language, and multiplies these into the $\overline{X}$ skeletons: choice of specifier, what a possible empty category can be in a particular language. To summarize, the following information is precompiled in Dorr's system:

- The $\overline{X}$ order for a language.
- Specifier choice (for example, determiner for $\overline{\overline{N}}$ in Spanish).
- The adjunction possibilities (for example, clitic *le* can adjoin to $\overline{V}$ in Spanish).
- Default values for nonlexical heads (for example, $\overline{\overline{V}}$ complement for I(nflection) in Spanish) (Here, Inflection simply means an element bearing tense, like the auxiliary verb *will* or *would* in English.)

To get the actual context-free rules for Spanish, we simply instantiate the $\overline{X}$ template with values for X={N, V, P, A } *at the time parsing occurs* plus two rules for I(nflections) and C(omplementizers), following details that are irrelevant here. For Spanish, the system knows that the choice of specifier for $\overline{\overline{C}}$ may be a wh-phrase like *which* or *what*; and it also knows that if a COMP head is absent, then the complement of C is $\overline{\overline{I}}$. Finally, it knows that a V(erb) and $\overline{\overline{N}}$ 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 $\overline{X}$ skeleton system multiplies out to on the order of a hundred or so context-free rules.

Dorr's full parser works by using the $\overline{X}$ skeleton rules as a driver program coroutined with all remaining principles, as sketched in table 2. The $\overline{X}$ component vastly overgenerates; it builds underspecified phrase structure because it does not access details of the thematic roles each verb demands. The actual parser is simply an Earley parser, using the context-free rules given by $\overline{X}$ expansion. As each word is processed, the parser uses the standard Earley algorithm PUSH, SCAN, or POP actions until no more actions trigger. It then calls on the remaining principles like the Case Filter or the Theta Criterion. These principles also call 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 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 principle module component has three tasks, accessed on demand: first, it weeds out bad parses (for example, a parse that calls for a complement when the verb does not need one or if bounding conditions are violated); second, it tries out possibilities that the $\overline{X}$ skeletal parser does not know about (for example, it can PREDICT that an empty category

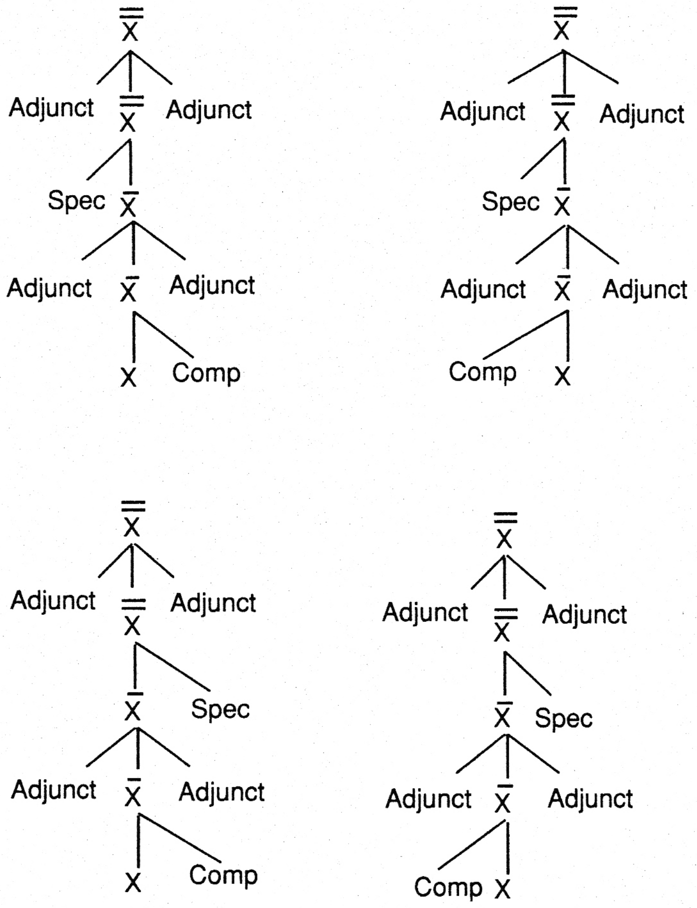


Figure 9. Dorr's $\overline{X}$ theory provides for four basic $\overline{X}$ skeleton structures on which to base further computation.

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).

Table 2 summarizes how work is split between the $\overline{X}$ component and the remaining principles. The overall effect is to implement a back-and-forth flow of information between the Earley parser actions and the 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 $\overline{X}$ and constraint modules. For example, Spanish permits $\overline{\overline{N}}$, *wh*-phrase, V(erb), I(nflection), *have*-aux, and *be*-aux as empty categories; while English permits only $\overline{\overline{N}}$ and *wh*-phrase. These are used for predicting an empty category at PUSH time.

To see how this all fits together, consider the simple sentence *¿Qué vio?*

ACTION	Earley parser	Principle Constraint Module
PUSH	Expand nonterminal	Predict empty category Check empty category and bounding
SCAN	Traverse terminal	Determine argument structure Perform feature matching
POP	Complete nonterminal	Assign thematic roles Check theta criterion

Table 2. The Earley parser works in tandem with principle-based constraints in Dorr's parsing design. Each standard parser action (start a phrase, scan a token, complete a phrase) runs one or more different constraint checks or structure-building routines.

("What did she/he see"). Figures 10 and 11 show the parsing sequence and final results. There are four final generated trees, labeled (a)–(d), of which just one, (b), is valid. The figure carries this labeling through several steps to show where each tree comes from.

As mentioned, this sentence exhibits three interesting Spanish syntactic phenomena: a null subject; inversion of the verb; and movement of *qué* to the front of the sentence. All of this can be captured *without* explicit rules.

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 parse itself then begins.

In step 1 as shown in figure 10, the parser first accesses the $\overline{\mathrm{X}}$ skeleton for $\overline{\overline{\mathrm{C}}}$, which is simply C-spec–C–C-comp (in Spanish), and where C need not be overt. A precompiled parameter particular for Spanish forces selection of $\overline{\overline{\mathrm{I}}}$ to be the complement of C, because there is no head present. *Qué* is then scanned (attached) to $\overline{\overline{\mathrm{C}}}$ as its (optional) specifier (note once again that this may lead to overgeneration because no conditions are checked at this point to determine whether this attachment is in fact correct).

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

First, $\overline{\overline{\mathrm{I}}}$ may be expanded as V–$\overline{\overline{\mathrm{I}}}$ (with V filling the adjunct slot of $\overline{\overline{\mathrm{I}}}$); 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 $\overline{\overline{\mathrm{X}}}$ level.) Second, $\overline{\overline{\mathrm{I}}}$ may be expanded to contain $\overline{\overline{\mathrm{N}}}$ in I-spec (parse c, step 1.) At this point, the parser has access to the next word, *ver*, a verb. This rules out any expansion of $\overline{\overline{\mathrm{N}}}$ as its usual spec-head-comp sequence. For this parse then, no further $\overline{\mathrm{X}}$ actions can occur and the parser consults the principle constraint module.

Moving on to step 2 of the parse, the constraint module determines that the next symbol to operate on for parse (c) is a nonterminal; hence, a PUSH is required. The corresponding action is to predict an empty category. There are two types of empty categories in the system: a *trace*, the position of a moved noun phrase, and *pro*, an empty pronominal. Both are predicted here: there is a possible antecedent that is not too far away, and the subject position (first $\overline{\overline{\mathrm{N}}}$ under $\overline{\overline{\mathrm{I}}}$) is still open. (Several principles come into play here, notably bounding; 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 one special kind of empty category can be placed in the relevant position.) The principle constraint module has nothing to say about parse possibility 1, and parsing is returned to the $\overline{\mathrm{X}}$ component.

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

We proceed to step 3. The parser next scans *ver*. It can be attached to the V(erb) slot for parse (3a). The constraint module is then called upon: its job is to determine the argument structure for the verb *ver*. The lexical entry for *ver* predicts an $\overline{\overline{\mathrm{N}}}$ complement.[2] For parses (c) and (d) this prediction may be realized only as an empty category trace because there is no more input, linked to the antecedent empty category; for parse (a), however, the subject position, known to be (precompiled as) $\overline{\overline{\mathrm{N}}}$, is still open, so either two possible empty category types 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 specifier position for I, but we will not cover these here.)

All words are now scanned, so the principle constraint 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 because two semantic roles are assigned to the object position: the object is both goal and also, via its antecedent linking, agent. These parses are thrown out.

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

For parses (a) and (b), additional actions are possible (steps 5–7). The complement of $\bar{\bar{\mathrm{I}}}$ is precompiled as a $\bar{\bar{\mathrm{V}}}$ (as dictated by a precompiled parameter, not a rule, because I is empty) (step 5). Because there is no word present, this $\bar{\bar{\mathrm{V}}}$ may be expanded (in Spanish) by a precompiled $\bar{\mathrm{X}}$ template as an empty verb (because verbs may be traces), linked to *ver* plus (via online accessed subcategorization facts because of linking to *ver*) an $\bar{\bar{\mathrm{N}}}$ complement (steps 5 and 6, parses (a) and (b)). Finally, the constraint module must again be accessed at this point, because the parser has just predicted an $\bar{\bar{\mathrm{N}}}$ complement. Constraint principles dictate that the $\bar{\bar{\mathrm{N}}}$ may be realized only as an empty category trace in this position, and the results are shown in 7(a) and 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 herself/himself".

Note that of the remaining two possible parses, (a) and (b), (a) is ruled out by feature checking, because *ver* demands an animate subject. This leaves only parse (b) as valid, "What did she/he see" (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 filtering principles guiding the $\bar{\mathrm{X}}$ parse, rather than by particular rules.

The Computational Implementation of Principle-Based Parsers

We have now described two different ways to organize a principle-based system for parsing. Kashket's parser operated in a bottom-up, serial fashion; Dorr's coroutined $\bar{\mathrm{X}}$ parsing with constraint filtering. But many other architectures are possible. Are any of these best for all natural languages? For any natural language?

In an attempt to examine parser architecture more systematically, in this section we shall draw on the work of Fong [1989]. In particular, Fong has studied the problem of reordering principles to avoid doing unnecessary work. We will see that although a globally optimal strategy of principle ordering is impossible, it is possible to use standard heuristic techniques for conjunct ordering developed in other domains of AI to reduce parsing time by up to an order of magnitude. Interestingly, this leads to a parser that is dynamically varied, according to the type of sentence it must process—this is in contrast to the usually fixed algorithmic regime of a rule-based system. The key questions about principle-ordering are these:

- What effect, if any, does principle-ordering have on the amount of work needed to parse a given sentence?

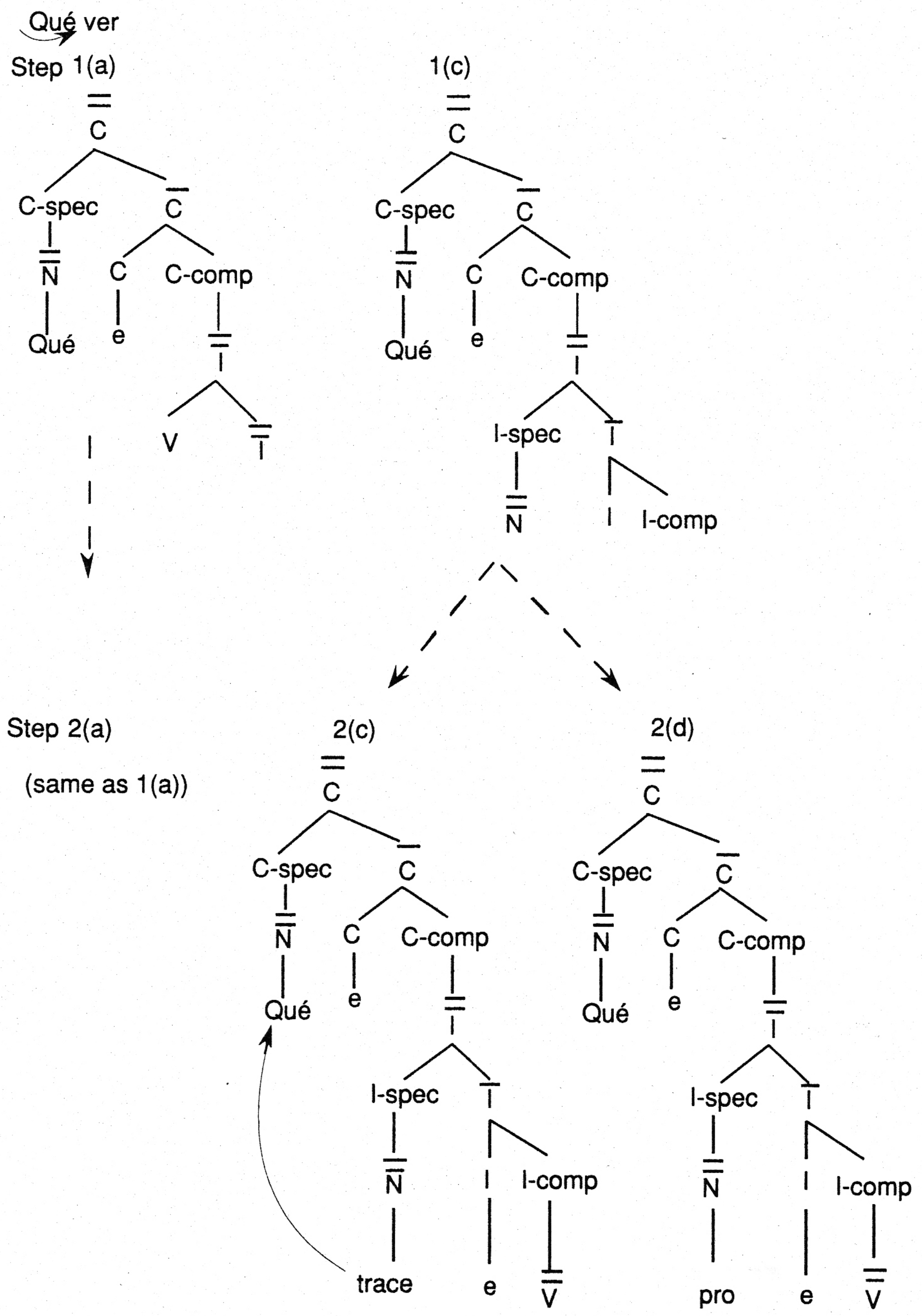


Figure 10. Beginning of the parse of *¿Qué vio?* in Dorr's system. This simple Spanish sentence has four possible parses, but only one is valid. This figure shows parsing steps 1–2. Parsing steps 3–7 are continued onto the next figures. (There are no trees 1(b)–3(b) or 1(d) because possibilities (b) and (d) will arise from the splitting of tree hypotheses in later steps. The *e* nodes stand for absent nodes, not empty categories.)

¿Qué vio?

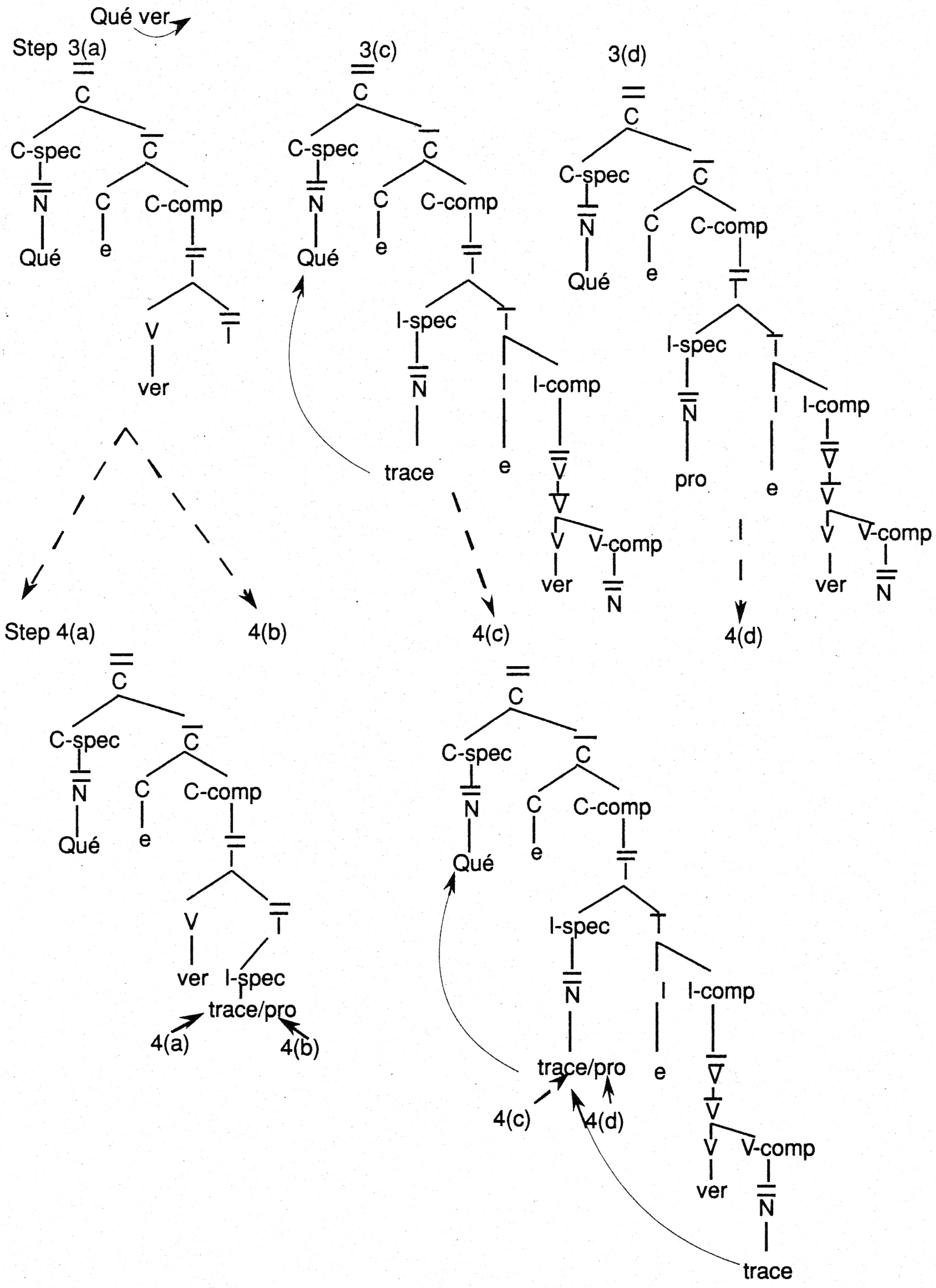
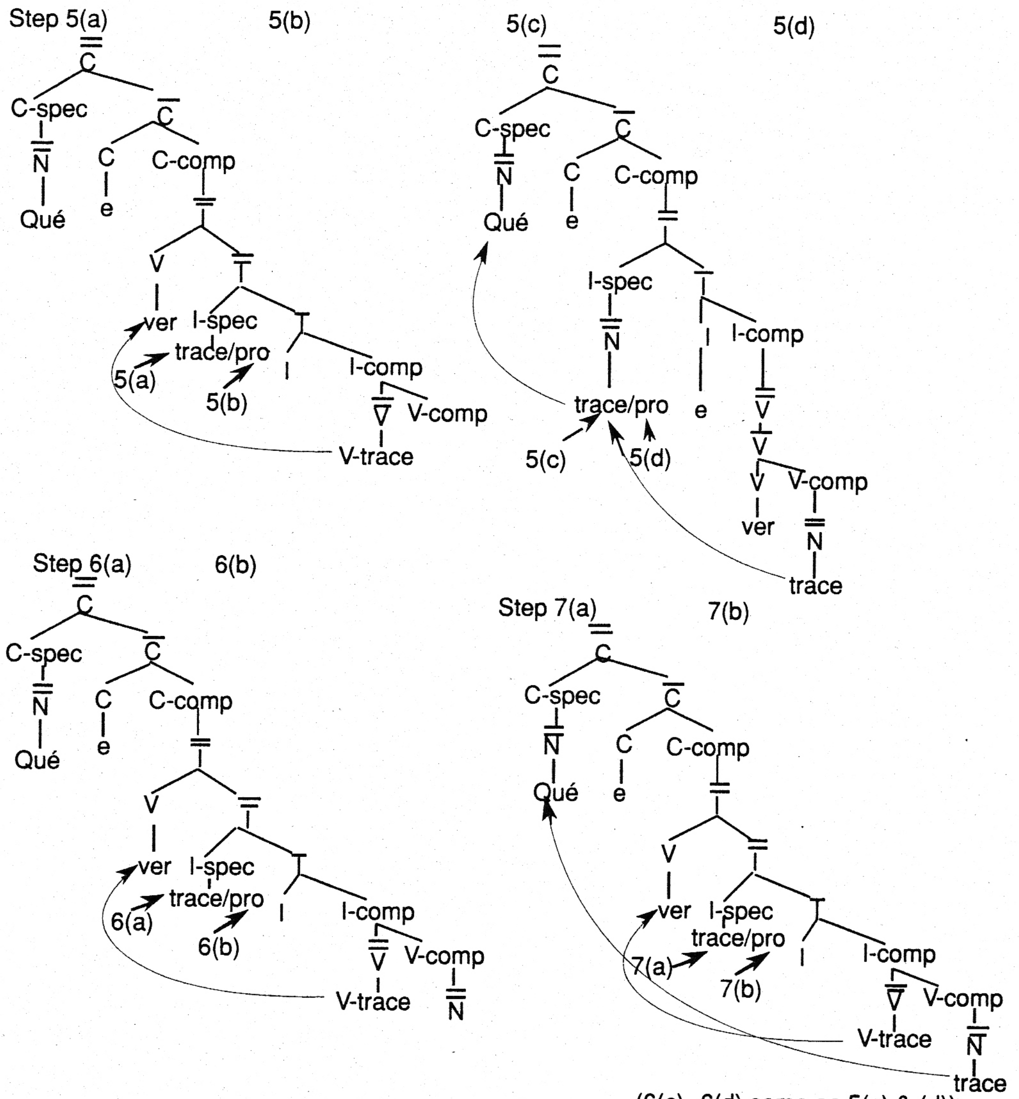


Figure 11. Continuation of the previous parse of *¿Qué vio?*, steps 3–4. This simple Spanish sentence has four possible parses, but only one is valid. Parses 4(c) and 4(d) remain the same through the next steps 5–7.



(6(c)--6(d) same as 5(c) & (d))

(7(c)--7(d) same as 5(c) & (d))

7(c)= What did what see itself

7(d)= What did she/he see herself/himself

Figure 12. Conclusion of the *¿Qué vio?* parse, steps 5–7. Only parse 7(b) passes all the conditions, corresponding to *What did she/he see?*. Parse 7(a) corresponds to the ill-formed *What saw*. Parses (c)–(d) are the same as they were in step 4, and correspond to the ill-formed sentences *What did what see itself* and *What did she/he see herself/himself*.

- If the effect of principle-ordering is significant, then are some orderings much better than others?
- If so, is it possible to predict (and explain) which ones these are?

A novel, logic-based parser, the Principle-Ordering Parser (POP) has been built by Fong to investigate and demonstrate the effects of principle-ordering. The POP was deliberately constructed in a highly modular fashion to allow for maximum flexibility in exploring alternative orderings of principles. For instance, each principle is represented separately as an atomic parser operation. A structure is deemed to be well-formed only if it passes all parser operations. The scheduling of parser operations is controlled by a dynamic ordering mechanism that attempts to eliminate unnecessary work by eliminating ill-formed structures as quickly as possible. (For comparison purposes, the POP also allows the user to turn off the dynamic ordering mechanism and to parse with a user-specified (fixed) sequence of operations.)

Although it was designed primarily to explore the computational properties of principles for building more efficient parsers, this parser is also designed to be capable of handling a reasonably wide variety of linguistic phenomena. The system faithfully implements most of the principles contained in Lasnik and Uriagereka [1988]. That is, the parser makes the same grammaticality judgments and reports the same violations for ill-formed structures as the reference text. Some additional theory is also drawn from Chomsky [1981, 1986]. Parser operations implement principles from Theta Theory, Case Theory, Binding Theory, Locality theory, and the Empty Category Principle, as described earlier in figure 2.

The principle ordering problem

How important an issue is the principle ordering problem in parsing? An informal experiment was conducted using the parser described in the previous section to provide some indication on the magnitude of the problem. Although we were unable to examine all the possible orderings, it turns out that order-of-magnitude variations in parsing times could be achieved merely by picking a few sample orderings.[3]

Explaining the variation in principle ordering

The variation in parsing times for various principle orderings that was observed can be explained by assuming that overgeneration is the main

[3] The parser has about twelve to sixteen modules. Given a set of one dozen operations, there are about 500 million different ways to order these operations. Fortunately, only about half a million of these are actually valid, due to logical dependencies between the various operations. However, this is still far too many to test exhaustively. Instead, only a few well-chosen orderings were tested on a number of sentences from the reference. The procedure involved choosing a default sequence of operations and 'scrambling' the sequence by moving operations as far as possible from their original positions (modulo any logical dependencies between operations).

bottleneck for the parser. That is, in the course of parsing a single sentence, a parser will hypothesize many different structures. Most of these structures, the ill-formed ones in particular, will be accounted for by one or more linguistic filters. A sentence will be deemed acceptable if there exists one or more structures that satisfy every applicable filter. Note that even when parsing *grammatical* sentences overgeneration will produce ill-formed structures that need to be ruled out. Given that our goal is to minimize the amount of work performed during the parsing process, we would expect a parse using an ordering that requires the parser to perform extra work, compared with another ordering, to be slower.

Overgeneration implies that we should order principle filters to eliminate ill-formed structures as quickly as possible. For these structures, applying any parser operation other than one that rules it out may be considered as doing extra, or unnecessary, work (modulo any logical dependencies between principles).

Global optimal ordering is impossible

Because some orderings perform better than others, a natural question to ask is: Does there exist a globally optimal ordering? The existence of such an ordering would have important implications for the design of the control structure of any principle-based parser. The parser has a novel dynamic control structure in the sense that it tries to determine an ordering-efficient strategy for every structure generated. If such a globally optimal ordering could be found, then we can do away with the run-time overhead and parser machinery associated with calculating individual orderings. That is, we could build an ordering-efficient parser simply by hardwiring the optimal ordering into its control structure. Unfortunately, no such ordering can exist.

It is easy to see why this is so. The impossibility of the globally optimal ordering follows directly from the "eliminate unnecessary work" ethic. Computationally speaking, an optimal ordering is one that rules out ill-formed structures at the earliest possible opportunity. A *globally* optimal ordering would be one that always ruled out every possible ill-formed structure without doing any unnecessary work. Consider the following three structures (taken from Lasnik and Uriagereka [1988]):

(a) *John$_1$ is crucial [$_{CP}$[$_{IP}$ e_1 to see this]]
(b) *[$_{NP}$John$_1$'s mother][$_{VP}$ likes himself$_1$]
(c) *John$_1$ seems that he$_1$ likes e_1

Example (a) violates the Empty Category Principle (ECP). Hence the optimal ordering must invoke the ECP operation before any other operation that it is not dependent on. On the other hand, example (b) violates a

Binding Theory principle. Hence, the optimal ordering must also invoke this principle as early as possible. Given that the two operations are independent, the optimal ordering must order the binding principle before the ECP and vice-versa. Similarly, example (c) demands that a variant of the Case filter must precede the other two operations. Hence a globally optimal ordering is impossible.

Heuristics for principle ordering

While one cannot achieve a globally optimal ordering, it is still possible to apply heuristic strategies that often come close. The principle-ordering problem can be viewed as a limited instance of the well-known conjunct ordering problem [Smith & Genesereth 1985]. Given a set of conjuncts, we are interested in finding all solutions that satisfy all the conjuncts simultaneously. The parsing problem is then to find well-formed structures (solutions) that satisfy all the parser operations (conjuncts) simultaneously. Moreover, we are particularly interested in minimizing the cost of finding these structures by re-ordering the set of parser operations.

This section outlines some of the heuristics used by Fong's parser to determine the minimum cost ordering for a given structure. The parser contains a dynamic ordering mechanism that attempts to compute a minimum cost ordering for every phrase structure generated during the parsing process.

This heuristic mechanism can be subdivided into two distinct phases which are discussed in turn. First, the dynamic ordering mechanism decides which principle is the most likely candidate for eliminating a given structure. Then the parser makes use of this information to reorder parser operation sequences to minimize the total work performed.

Predicting failing filters

Given any structure, the dynamic ordering mechanism attempts to satisfy the "eliminate unnecessary work" ethic by predicting a "failing" filter for that structure. More precisely, it will try to predict the principle that a given structure violates on the basis of the simple structure cues. Because the ordering mechanism cannot know whether a structure is well-formed or not, it assumes that all structures are ill-formed and attempts to predict a failing filter for every structure. In order to minimize the amount of work involved, the types of cues that the dynamic ordering mechanism can test for are deliberately limited. Only inexpensive tests such as whether a category contains certain features are used. Any cues that may require significant computation, such as searching for an antecedent, are considered to be too expensive. Each structure cue is then associated with a list of possible failing filters. (Some examples of the mapping between cues and filters are shown in table 3 below.) The system then chooses one of the possible failing filters based on this mapping.

Structure cue	*Possible failing filters*
trace	Empty Category Principle, and Case Condition on traces
intransitive	Case Filter
passive	Theta Criterion Case Filter
non-argument	Theta Criterion
+anaphoric	Binding Theory Principle A
+pronominal	Binding Theory Principle B

Table 3. Some of the heuristic cues used for testing which filter will block a given sentence.

The correspondence between each cue and the set of candidate filters may be systematically derived from the definitions of the relevant principles. For example, Principle A of the Binding Theory deals with the conditions under which antecedents for anaphoric items such as *each other* and *himself* must appear. Hence, Principle A can only be a candidate failing filter for structures that contain an item with the +anaphoric feature. Other correspondences may be somewhat less direct. For example, the Case Filter merely states that all overt noun phrases must have abstract Case. Now, in the parser the conditions under which a noun phrase may receive abstract Case are defined by two separate operations, namely, Inherent Case Assignment and Structural Case Assignment. It turns out that an instance where Structural Case Assignment will not assign Case is when a verb that normally assigns Case has passive morphology. Hence, the presence of a passive verb in a given structure may cause an overt noun phrase to fail to receive Case during Structural Case Assignment—which in turn may cause the Case Filter to fail.[4]

[4] It is possible to automate the process of finding structure cues simply by inspecting the closure of the definitions of each filter and all dependent operations. One method of deriving cues is to collect the negation of all conditions involving category features. For example, if an operation contains the condition "`not (Item has_feature intransitive)`", then we can take the presence of an intransitive item as a possible reason for failure of that operation. However, this approach has the potential problem of generating too many cues. Although, it may be relatively inexpensive to test each individual cue, a large number of cues will significantly increase the overhead of the ordering mechanism. Furthermore, it turns out that not all cues are equally useful in predict-

The failing filter mechanism can be seen as an approximation to the Cheapest-first heuristic in conjunct ordering problems. It turns out that if the cheapest conjunct at any given point will reduce the search space rather than expand it, then it can be shown that the optimal ordering must contain that conjunct at that point. Obviously, a failing filter is a "cheapest" operation in the sense that it immediately eliminates one structure from the set of possible structures under consideration.

Although the dynamic ordering mechanism performs well in many of the test cases drawn from the reference text, it is by no means foolproof. There are also many cases where the prediction mechanism triggers an unprofitable re-ordering of the default order of operations. (We will present one example of this in the next section.)

Logical dependencies and reordering

Given a candidate failing filter, the dynamic ordering mechanism has to schedule the sequence of parser operations so that the failing filter is performed as early as possible. Simply moving the failing filter to the front of the operations queue is not a workable approach for two reasons.

First, simply fronting the failing filter may violate logical dependencies between various parser operations. For example, suppose the Case Filter were chosen to be the failing filter. To create the conditions under which the Case Filter can apply, both Case assignment operations, namely, Inherent Case Assignment and Structural Case Assignment, must be applied first. Hence, fronting the Case Filter will also be accompanied by the subsequent fronting of both assignment operations—unless they have already been applied to the structure in question.

Second, the failing filter approach does not take into account the behavior of generator principles. Due to logical dependencies it may be necessary in some situations to invoke a generator operation before a failure filter can be applied. For example, the filter Principle A of the Binding Theory is logically dependent on the generator Free Indexing to generate the possible antecedents for the anaphors in a structure. Consider the possible binders for the anaphor *himself* in *John thought that Bill saw himself* as shown below:

ing failure filters. One solution may be to use "weights" to rank the predictive utility of each cue with respect to each filter. Then an adaptive algorithm could be used to "learn" the weighting values, in a manner reminiscent of Samuels [1967]. The failure filter prediction process could then automatically eliminate testing for relatively unimportant cues. This approach is currently being investigated.

(a) *John$_i$ thought that Bill$_j$ saw himself$_i$

(b) John$_i$ thought that Bill$_j$ saw himself$_j$

(c) *John$_i$ thought that Bill$_j$ saw himself$_k$

Only in example (b) is the antecedent close enough to satisfy the locality restrictions imposed by Principle A. Note that Principle A had to be applied a total of three times in the above example in order to show that there is only one possible antecedent for *himself*. This situation arises because of the general tendency of generators to overgenerate. But this characteristic behavior of generators can greatly magnify the extra work that the parser does when the dynamic ordering mechanism picks the wrong failing filter. Consider the ill-formed structure **John seems that he likes e* (a violation of the principle that traces of noun phrases cannot receive Case). If however, Principle B of the Binding Theory is predicted to be the failure filter (on the basis of the structure cue *he*), then Principle B will be applied repeatedly to the possibilities generated by free indexing. On the other hand, if the Case Condition on Traces operation was correctly predicted to be the failing filter, then Free Indexing need not be applied at all. The dynamic ordering mechanism of the parser is designed to be sensitive to the potential problems caused by selecting a candidate failing filter that is logically dependent on many generators.

The utility of principle ordering

From Fong's experiments with the parser we have found that dynamic principle-ordering can provide a significant improvement over any fixed ordering. We have found that speed-ups varying from three- or four-fold to order-of-magnitude improvements are possible in many cases.[5]

The control structure of the current parser forces linguistic principles to be applied one at a time. Many other machine architectures are certainly possible, and will be explored in future research. For example, we could take advantage of the independence of many principles and apply principles in parallel whenever possible. However, any improvement in parsing performance would come at the expense of violating the minimum (unnecessary) work ethic. Lazy evaluation of principles is yet another alternative. However, principle-ordering would still be an important consideration for efficient processing in this case. Finally, we should also consider principle-ordering from the viewpoint of scalability. The experience with prototypes suggests that as the level of sophistication of the parser increases (both in

[5] Obviously the speed-up obtained will depend on the number of principles present in the system and the degree of fine-tuning of the failure filter selection criteria.

terms of the number and complexity of individual principles), the effect of principle-ordering also becomes more pronounced.

References

Abney, S. [1986], "A new model of natural language parsing," unpublished M.S. Thesis, Department of Linguistics and Philosophy, Massachusetts Institute of Technology, Cambridge, MA.

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

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

Chomsky, N. [1986], *Knowledge of Language: Its Nature, Origin, and Use*, Prager.

Dorr, B. J. [1987], "UNITRAN: A Principle-Base Approach to Machine Translation," Report AI-TR-1000, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Fong, S. [to appear 1989], "Principle-based parsing and Principle-Ordering," Report AIM-1156, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Johnson, M. [1988], "Parsing as deduction: the use of knowledge of language." Knowledge as Language," In *The MIT Parsing Volume*, edited by S. Abney, Center for Cognitive Science, Cambridge, MA, pp. 47-69.

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

Kartunnen, L., and M. Kay [1985], in *Natural Language Parsing*, edited by Dowty, Kartunnen, and Zwicky, Cambridge University Press, Cambridge, England, pp. 279–306.

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

Laughren, M. [1987], "The configurationality parameter and Warlpiri," in *Configurationality*, edited by L. Maracz and Pl. Muysken, Dordrecht, Foris Publications.

Samuels, A. L. [1967], "Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress," *IBM Journal.*

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

Work on principle-based parsing at the Artificial Intelligence Laboratory has been supported by NSF Grant DCR-85552543 under a Presidential Young Investigator's Award to Professor Robert C. Berwick, by a grant from the Lotus Development Corporation, by the Kapor Family Foundation, and by the IBM Graduate Research Fellowship Program.

Smith, D. E., and M. R. Genesereth [1985], "Ordering Conjunctive Queries," *Artificial Intelligence* vol. 26, pp. 171–215.

Stabler, E. P., Jr. [1989] "The Logical Approach to Syntax: Foundations, Specifications and Implementations of Theories of Government and Binding," *m.s.*, London, Ontario, University of Western Ontario.

Tenney, C. [1986], "A context-free rule system for parsing Japanese," *MIT-Japan Science and Technology Program*, Cambridge, MA.