SYNTACTIC CONSTRAINTS AND EFFICIENT PARSABILITY

Robert C. Berwick
Room 820, MIT Artificial Intelligence Laboratory
545 Technology Square, Cambridge, MA 02139
Amy S. Weinberg
Department of Linguistics, MIT
Cambridge, MA 02139

ABSTRACT

A central goal of linguistic theory is to explain why natural languages are the way they are. It has often been supposed that computational considerations ought to play a role in this characterization, but rigorous arguments along these lines have been difficult to come by. In this paper we show how a key "axiom" of certain theories of grammar, Subjacency, can be explained by appealing to general restrictions on on-line parsing plus natural constraints on the rule-writing vocabulary of grammars. The explanation avoids the problems with Marcus' [1980] attempt to account for the same constraint. The argument is robust with respect to machine implementation, and thus avoids the problems that often arise when making detailed claims about parsing efficiency. It has the added virtue of unifying in the functional domain of parsing certain grammatically disparate phenomena, as well as making a strong claim about the way in which the grammar is actually embedded into an on-line sentence processor.

I INTRODUCTION

In its short history, computational linguistics has been driven by two distinct but interrelated goals. On the one hand, it has aimed at computational explanations of distinctively human linguistic behavior -- that is, accounts of why natural languages are the way they are viewed from the perspective of computation. On the other hand, it has accumulated a stock of engineering methods for building machines to deal with natural (and artificial) languages. Sometimes a single body of research has combined both goals. This was true of the work of Marcus [1980], for example. But all too often the goals have remained opposed -- even to the extent that current transformational theory has been disparaged as hopelessly "intractable" and no help at all in constructing working parsers.

This paper shows that modern transformational grammar (the "Government-Binding" or "GB" theory as described in Chomsky [1981]) can contribute to both aims of computational linguistics. We show that by combining simple assumptions about efficient parsability along with some assumptions about just how grammatical theory is to be "embedded" in a model of language processing, one can actually explain some key constraints of natural languages, such as Subjacency. (The argument is different from that used in Marcus [1980].) In fact, almost the entire pattern of constraints taken as "axioms" by the GB theory can be accounted for. Second, contrary to what has sometimes been supposed, by exploiting these constraints we can show that a GB-based theory is particularly compatible with efficient parsing designs, in particular, with extended LR(k,t) parsers (of the sort described by Marcus [1980]). We can extend the LR(k,t) design to accommodate such phenomena as antecedent-PRO and pronominal binding, rightward movement, gapping, and VP deletion.

A. Functional Explanations of Locality Principles

Let us consider how to explain locality constraints in natural languages. First of all, what exactly do we mean by a "locality constraint"? The paradigm case is that of Subjacency: the distance between a displaced constituent and its "underlying" canonical argument position cannot be too large, where the distance is gauged (in English) in terms of the number of the number of S(entence) or NP phrase boundaries. For example, in sentence (1a) below, *John* (the so-called "antecedent") is just one S-boundary away from its presumably "underlying" argument position (denoted "x", the "trace")) as the Subject of the embedded clause, and the sentence is fine:

(1a) John seems $[_S$ x to like ice cream].

However, all we have to do is to make the link between *John* and *x* extend over two S's, and the sentence is ill-formed:

(1b) John seems $[_S$ it is certain $[_S$ x to like ice cream

This restriction entails a "successive cyclic" analysis of transformational rules (see Chomsky [1973]). In order to derive a sentence like (1c) below without violating the Subjacency condition, we must move the NP from its canonical argument position through the empty Subject position in the next higher S and then to its surface slot:

(1c) John seems [e] to be certain x to get the ice cream.

Since the intermediate subject position is filled in (1b) there is no licit derivation for this sentence.

More precisely, we can state the Subjacency constraint as follows:

No rule of grammar can involve X and Y in a configuration like the following,

$$[_\epsilon \ldots X \ldots [_\alpha \ldots [_\beta \ldots Y \ldots] \ldots] \ldots X \ldots]$$

where α and β are bounding nodes (in English, S or NP phrases).

Why should natural languages be designed this way and not some other way? Why, that is, should a constraint like Subjacency exist at all? Our general result is that under a certain set of assumptions about grammars and their relationship to human sentence processing one can actually expect the following pattern of syntactic locality constraints:

> (1) The antecedent-trace relationship *must* obey Subjacency, but other "binding" realtionships (e.g., NP--Pro) need not obey Subjacency.

(2) Gapping constructions *must* be subject to a bounding condition resembling Subjacency, but VP deletion need not be.

(3) Rightward movement *must* be strictly bounded.

To the extent that this predicted pattern of constraints is actually observed -- as it is in English and other languages -- we obtain a genuine functional explanation of these constraints and support for the assumptions themselves. The argument is different from Marcus' because it accounts for syntactic locality constraints (like Subjacency) as the *joint* effect of a particular theory of grammar, a theory of how that grammar is used in parsing, a criterion for efficient parsability, and a theory of of how the parser is built. In contrast, Marcus attempted to argue that Subjacency could be derived from just the (independently justified) operating principles of a particular kind of parser.

B. Assumptions.

The assumptions we make are the following:

(1) The grammar includes a level of annotated surface structure indicating how constituents have been displaced from their canonical predicate argument positions. Further, sentence analysis is divided into two stages, along the lines indicated by the theory of Government and Binding: the first stage is a purely syntactic analysis that rebuilds annotated surface structure; the second stage carries out the interpretation of variables, binds them to operators, all making use of the "referential indices" of NPs.

(2) To be "visible" at a stage of analysis a linguistic representation must be written in the vocabulary of that level. For example, to be affected by syntactic operations, a representation must be expressed in a syntactic vocabulary (in the usual sense); to be interpreted by operations at the second stage, the NPs in a representation must possess referential indices. (This assumption is not needed to derive the Subjacency constraint, but may be used to account for another "axiom" of current grammatical theory, the so-called "constituent command" constraint on antecedents and the variables that they bind.) This "visibility" assumption is a rather natural one.

(3) The rule-writing vocabulary of the grammar cannot make use of arithmetic predicates such as "one", "two" or "three", but only such predicates as "adjacent". Further, quantificational statements are not allowed in rules. These two assumptions are also rather standard. It has often been noted that grammars "do not count" -- that grammatical predicates are structurally based. There is no rule of grammar that takes the just the fourth constituent of a sentence and moves it, for example. In contrast, many different kinds of rules of grammar make reference to adjacent constituents. (This is a feature found in morphological, phonological, and syntactic rules.)

(4) Parsing is not done via a method that carries along (a representation) of all possible derivations in parallel. In particular, an Earley-type algorithm is ruled out. To the extent that multiple options about derivations are *not* pursued, the parse is "deterministic."

(5) The left-context of the parse (as defined in Aho and Ullman [1972]) is *literally* represented, rather than generatively represented (as, e.g., a regular set). In particular, just the symbols used by the grammar (S, NP, VP...) are part of the left-context vocabulary, and not "complex" symbols serving as proxies for the set of left-context strings.[1] In effect, we make the (quite strong) assumption that the sentence processor adopts a direct, transparent embedding of the grammar.

Other theories or parsing methods do not meet these constraints and fail to explain the existence of locality constraints with respect to this particular set of assumptions. [2] For example, as we show, there is no reason to *expect* a constraint like Subjacency in the Generalized Phrase Structure Grammars (GPSGs) of Gazdar [1981], because there is no inherent barrier to easily processing a sentence where an antecedent and a trace are unboundedly far from each other. Similarly, if a parsing method like Earley's algorithm were actually used by people, then Subjacency remains a mystery on the functional grounds of efficient parsability. (It could still be explained on other functional grounds, e.g., that of learnability.)

II PARSING AND LOCALITY PRINCIPLES

To begin the actual argument then, assume that on-line sentence processing is done by something like a deterministic parser.[3] Sentences like (2) cause trouble for such a parser:

(2) What $_i$ do you think that John told Mary...that he would like to eat e_i.

1. Recall that the successive lines of a left- or right-most derivation in a context-free grammar constitute a regular language, as shown in, e.g., DeRemer [1969].

2. Plainly, one is free to imagine some *other* set of assumptions that would do the job.

3. If one assumes a backtracking parser, then the argument can also be made to go through, but only by assuming that backtracking is very costly. Since this sort of parser clearly subsumes the LR(k)-type machines under the right construal of 'cost', we make the stronger assumption of LR(k)-ness.

The problem is that on recognizing the verb *eat* the parser must decide whether to expand the parse with a trace (the transitive reading) or with no postverbal element (the intransitive reading). The ambiguity cannot be locally resolved since *eat* takes both readings. It can only be resolved by checking to see whether there is an actual antecedent. Further, observe that this is indeed a *parsing* decision: the machine must make some decision about how to to build a portion of the parse tree. Finally, given non-parallelism, the parser is not allowed to pursue both paths at once: it must decide now how to build the parse tree (by inserting an empty NP trace or not).

Therefore, assuming that the correct decision is to be made on-line (or that retractions of incorrect decisions are costly) there must be an actual parsing rule that expands a category as transitive iff there is an immediate postverbal NP in the string (no movement) or if an actual antecedent is present. However, the phonologically overt antecedent can be unboundedly far away from the gap. Therefore, it would seem that the relevant parsing rule would have to refer to a potentially unbounded left context. Such a rule cannot be stated in the finite control table of an LR(k) parser. Therefore we must find some finite way of expressing the domain over which the antecedent must be searched.

There are two ways of accomplishing this. First, one could express all possible left-contexts as some regular set, and then carry this representation along in the finite control table of the LR(k) machine. This is always possible in the case of a context-free grammar, and in fact is the "standard" approach.[4] However, in the case of (e.g.) *wh* movement, this demands a generative encoding of the associated finite state automaton, via the use of complex symbols like "S/wh" (denoting the "state" that a *wh* has been encountered) and rules to pass long this non-literal representation of the state of the parse. This approach works, since we can pass along this state encoding through the VP (via the complex non-terminal symbol VP/wh) and finally into the embedded S. This complex non-terminal is then used to trigger an expansion of *eat* into its transitive form. In fact, this is precisely the solution method advocated by Gazdar. We see then that if one adopts a non-terminal encoding scheme there should be no problem in parsing any single long-distance gap-filler relationship. That is, there is no need for a constraint like Subjacency.[5]

Second, the problem of unbounded left-context is directly avoided if the search space is limited to some literally finite left context. But this is just what the Subjacency constraint does: it limits where an antecedent NP could be to an immediately adjacent S or $\bar{S}$. This constraint has a simple interpretation in an actual parser (like that built by Marcus [1980]). The IF-THEN pattern-action rules that make up the Marcus parser's finite control "transition table" must be finite in order to be stored inside a machine. The rule actions themselves are literally finite. If the rule patterns must be *literally* stored (e.g., the pattern $[_S\ [_S ... [_S$ must be stored as an actual arbitrarily long string of S nodes, rather than as the regular set S^+), then these patterns must be literally finite. That is, parsing patterns must refer to literally bounded right and left context (in terms of phrasal nodes).[6] Note further that this constraint depends on the sheer representability of the parser's rule system in a finite machine, rather than on any details of implementation. Therefore it will hold invariantly with respect to machine design -- no matter kind of machine we build, if we assume a literal representation of left-contexts, then some kind of finiteness constraint is required. The robustness of this result contrasts with the usual problems in applying "efficiency" results to explain grammatical constraints. These often fail because it is difficult to consider all possible implementations simultaneously. However, if the argument is invariant with respect to machine desing, this problem is avoided.

Given literal left-contexts and no (or costly) backtracking, the argument so far motivates *some* bounding condition for ambiguous sentences like these. However, to get the full range of cases these functional facts must interact with properties of the rule writing system as defined by the grammar. We will derive the fact that the bounding condition must be subjacency (as opposed to tri- or quad-jacency) by appeal to the fact that grammatical constraints and rules are stated in a vocabulary which is *non-counting*. Arithmetic predicates are forbidden. But this means that since only the predicate "adjacent" is permitted, any literal bounding restriction must be expressed in terms of adjacent domains; hence Subjacency. (Note that "adjacent" is also an arithmetic predicate.) Further, Subjacency must apply to all traces (not just traces of ambiguously transitive/intransitive verbs) because a restriction to just the ambiguous cases would involve using existential quantification. Quantificational predicates are barred in the rule writing vocabulary of natural grammars.[7]

Next we extend the approach to NP movement and Gapping. Gapping is particularly interesting because it is difficult to explain why this construction (unlike other deletion rules) is bounded. That is, why is (3) but not (4) grammatical:

(3) John will hit Frank and Bill will $[e]_{VP}$ George.

*(4)John will hit Frank and I don't believe Bill will $[e]_{VP}$George.

The problem with gapping constructions is that the attachment of phonologically identical complements is governed by the verb that the complement follows. Extraction tests show that in (5) the phrase *after Mary* attaches to V' while in (6) it attaches to V''. (See Hornstein and Weinberg [1981] for details.)

(5) John will run after Mary.
(6) John will arrive after Mary.

In gapping structures, however, the verb of the gapped constituent is not present in the string. Therefore, correct attachment of the complement can only be guaranteed by accessing the antecedent in the previous clause. If this is true however, then the bounding argument for Subjacency applies to this case as well: given deterministic parsing of gapping done correctly, and a literal representation of left-context, then gapping must be context-bounded. Note that this is a particularly

4 Following the approach of DeRemer [1969], one builds a finite state automaton that recognizes exactly the set of left-context strings that can arise during the course of a right-most derivation, the so-called *characteristic finite state automaton*.

5 Plainly the same holds for a "hold cell" approach to computing filler-gap relationships.

6. Actually then, this kind of device falls into the category of bounded context parsing, as defined by Floyd [1964].

7 Of course, there is a another natural predicate that would produce a finite bound on rule context: if NP and trace had to be in the same S domain. Presumably, this is also an option that could get realized in some natural grammars; the resulting languages would not have overt movement outside of an S. Note that the natural predicates simply give the range of possible natural grammars, not those actually found.

The elimination of quantification predicates is supportable on grounds of acquisition.

interesting example because it shows how grammatically dissimilar operations like *wh*-movement and gapping can "fall together" in the functional domain of parsing.

NP-trace and gapping constructions contrast with antecedent/(pro)nominal binding, lexical anaphor relationships, and VP deletion. These last three do not obey Subjacency. For example, a Noun Phrase can be unboundedly far from a (phonologically empty) PRO, even in terms of

> John$_i$ thought it was certain that... [PRO$_i$ feeding himself] would be easy.

Note though that in these cases the expansion of the syntactic tree does not depend on the presence or absence of an antecedent. (Pro)nominals and lexical anaphors are phonologically realized in the string and can unambiguously tell the parser how to expand the tree. (After the tree is fully expanded the parser may search back to see whether the element is bound to an antecedent, but this is not a parsing decision.) VP deletion sites are also always locally detectable from the simple fact that every sentence requires a VP. The same argument applies to PRO. PRO is locally detectable as the only phonologically unrealized element that can appear in an ungoverned context, and the predicate "ungoverned" is local.[8] In short, there is no *parsing* decision that hinges on establishing the PRO-antecedent, VP deletion-antecedent, or lexical anaphor-antecedent relationship. But then, we should not expect bounding principles to apply in these cases, and, in fact, we do not find these elements subject to bounding. Once again then, apparently diverse grammatical phenomena behave alike within a functional realm.

To summarize, we can explain why Subjacency applies to exactly those elements that the grammar stipulates it must apply to. We do this using both facts about the functional design of a parsing system and properties of the formal rule writing vocabulary. To the extent that the array of assumptions about the grammar and parser actually explain this observed constraint on human linguistic behavior, we obtain a powerful argument that certain kinds of grammatical representations and parsing designs are actually implicated in human sentence processing.

III ACKNOWLEDGEMENTS

This report describes work 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 of the Department of Defense under Office of Naval Research Contract N00014-80-C-0505.

IV REFERENCES

Aho, Alfred and Ullman, Jeffrey [1972] *The Theory of Parsing, Translation, and Compiling*, vol. I., Prentice-Hall.

Chomsky, Noam [1973] "Conditions on Transformations,"in S. Anderson & P Kiparsky, eds. *A Festschrift for Morris Halle*. Holt, Rinehart and Winston.

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

DeRemer, Frederick [1969] *Practical Translators for LR(k) Languages*, PhD dissertation, MIT Department of Electrical Engineering and Computer Science.

Floyd, Robert [1964] "Bounded-context syntactic analysis," *Communications of the Association for Computing Machinery*, 7, pp. 62-66.

Gazdar, Gerald [1981] "Unbounded dependencies and coordinate structure," *Linguistic Inquiry*, 12:2 155-184.

Hornstein, Norbert and Weinberg, Amy [1981] "Preposition stranding and case theory," *Linguistic Inquiry*, 12:1.

Marcus, Mitchell [1980] *A Theory of Syntactic Recognition for Natural Language*, MIT Press

8 Since α is ungoverned iff α governed is false, and *α governed* is a bounded predicate, being restricted to roughly a single maximal projection (at worst an S).