Cross-Serial Dependencies Are Not Hard to Process

Carl Vogel
Institute for Computational Linguistics
University of Stuttgart
Azenbergstr. 12
D-70174 Stuttgart
Germany

Ulrike Hahn
Department of Experimental Psychology
University of Oxford
South Parks Road
Oxford OX1 3UD
England

Holly Branigan
Centre for Cognitive Science
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW
Scotland

{vogel,holly,ueh}@cogsci.ed.ac.uk

Abstract

Cross-serial dependencies in Dutch and Swiss-German are the only known extra-context free natural language syntactic phenomena. Psycholinguistic evidence suggests cross-serial orderings tend to be easier to process than nested constructions. We argue that the expressivity requirements of the corresponding formal languages do not actually entail that processing reduplication languages require the worst-case time complexity for languages of the same expressive class. We distinguish between context-free representability and context-free processing. We show that for any language with up to context free expressive power, *processing* cross-serial dependencies can be accommodated without affecting parsing complexity. This is related to other work on reduplication phenomena in formal models of computation.

1 Introduction

The cross-serial dependencies in Dutch and Swiss-German are the only known constituent-level syntactic phenomena which make natural languages not representable in context free languages (Gazdar, 1985; Gazdar and Pullum, 1985). Psycholinguistic study of the cross-serial dependencies reveals that the cross-serial orderings tend to be preferred over nested constructions (Bach et al., 1986).[1] Bach et al. argue from this that the pushdown stack cannot be the universal basis of the human parsing mechanism (since the pushdown automaton is essentially a context free recognition device which cannot represent cross-serial dependencies). Stabler (1994), on the other hand, considers the findings of Bach et al. (1986) as evidence for finite human sentence processing capacity. In this paper, we distinguish between context-free representability and context-free processing. We show that for any language with up to context free expressive power, *processing* cross-serial dependencies can be accommodated without affecting parsing complexity. While this does essentially imbue the language with indexed expressivity, it does so while allowing us to retain context free (or even regular) parsing complexity. Essentially, it is possible to carve out a cross-section of the expressivity hierarchy with the desired processing complexity. The result is based on the simple observation that the cross-serial dependencies are idealized by the string duplication language (whereas the nested dependencies are idealized by the palindrome language), and that it is trivial to provide a context-free (or regular) language parse for half of the string, followed by a test of equality for the remaining half of the string. This is consistent with findings that cross-serial dependencies are not hard to process, but qualifies the interpretation that Bach et al. give to their results and the implications on the human parsing mechanism. In particular, this suggests that with an additional operation the pushdown stack can be adequate for processing human languages. It also suggests an explanation for the finding that Dutch cross-serial dependencies are easier to process than German nested dependencies. We outline further consequences of our proposal in terms of patterns of disfluencies that are likely to occur in languages that admit cross-serial dependencies and propose a strategy for empirical investigation.

[1] Nested constructions are a quintessentially context free phenomenon.

2 Preliminaries

To calibrate our discussion, we quickly review the salient terminology from formal language theory and the current understanding of the import for natural languages.

2.1 Terminology

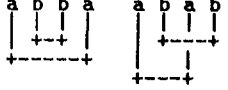
Let $\mathcal{L}i$ denote the hierarchy of languages generated by the corresponding hierarchy of grammars (according to the usual hierarchy (Hopcroft and Ullman, 1979)). Thus, $\mathcal{L}0$ denotes the class of languages generated by type 0 grammars. They are characterized by unrestricted grammar produc-

tion rules. $\mathcal{L}1$ is the class of languages generated by context sensitive grammars—the sole restriction on production rules in this type of grammar is that the right hand side (RHS) of each rule is at least as long as the left hand side (LHS). $\mathcal{L}1.5$ denotes the class of languages generated by indexed grammars. Gazdar (1985) provides the most perspicuous notation for the restricted forms that production rules may take in such grammars:[2]

1. $A[...] \longrightarrow W[...]$
2. $A[...] \longrightarrow B[i, ...]$
3. $A[i, ...] \longrightarrow W[...]$

Indexed grammars incorporate a notion of stacking; rules of the form in (2) describe push operations, and those of the form in (3) involve pops. Rules of the form (1) are copy operations. The elipses indicate that the remainder of the stack is passed on from the LHS to each nonterminal (and only the nonterminals) on the RHS. $\mathcal{L}2$ is the class of context free languages generated by grammars whose productions are restricted such that the LHS of each is a single nonterminal symbol, and each RHS is a sequence of terminals and nonterminals. Finally, the regular languages, $\mathcal{L}3$ are those produced by regular grammars, characterized by rules that have a single nonterminal symbol on the LHS and on the RHS, either a terminal symbol or a terminal and a single nonterminal.

These classes of languages can be arranged into a hierarchy based on proper containment relations among them: $\mathcal{L}3 \subset \mathcal{L}2 \subset \mathcal{L}1.5 \subset \mathcal{L}1 \subset \mathcal{L}0$ ($\mathcal{L}0$ is the least restrictive, the most expressive). Aho (1968) shows the existence of languages that are a proper subset of the indexed languages and a proper superset of the context free. Joshi et al. (1989) conjecture that there is actually a convergence in expressive power among the 'mildly context sensitive' (MCS) languages, but other work points out exceptions (Savitch, 1989; Vogel and Erjavec, 1994). Since the reduplication languages (Savitch, 1989) are central to the point of this paper we define them—the languages homomorphic to the set of strings $\{ww|w \in \{a,b\}^*\}$. The string duplication languages are not context free, although they are closely related to the string reversal languages ($\{ww^R|w \in \{a,b\}^*\}$, where the R indicates the reversal operator) which are context free. The two languages induce different dependency relationships which is best described as *nesting* in the context free case and *cross-serial* in the indexed case:



[2]The bracketed material indicates a stack of indices; W denotes a sequence of elements of terminals and nonterminals; A, B denote nonterminals.

An important property of the each of the language classes is that it is closed under both intersection with regular languages (e.g., the intersection of a context free language and a regular language is no more expressive than a context free language) and homomorphism (e.g., an order preserving map of each symbol in a language to a single element (possibly a string) of a context free language implies that the first language is also context free). It is convenient to refer to languages with homomorphisms to $\{ww^R|w \in \{a,b\}^*\}$ and $\{ww|w \in \{a,b\}^*\}$ as ww^R and ww, respectively.

Corresponding to expressivity class and the associated model of computation is the complexity of recognition for each class. Table 1 gives an informal ranking of the language classes with their corresponding worst case recognition complexity on the standard model of computation. Thus, given a context free grammar for ww^R and a string of length n, then in the worst case it will take an amount of time proportional to the cube of the length of the string to determine whether the string is in ww^R (and identify its structure). While the expressivity hierarchy is useful for differentiating classes of languages in precise terms like worst-case recognition complexity, it is easy to use the hierarchy incorrectly. For instance, it is not valid to conclude that because a language is in a particular language class all subsets of that language are also included that language class (e.g. ww^R is a proper subset of w, yet $w \in \mathcal{L}3$ $ww^R \in \mathcal{L}2$). Also, in most cases the structural descriptions that underlie strings of a language are of more interest than the string sets themselves. For this reason it is useful to distinguish *weak* and *strong* containment of a grammar in a language class: e.g., a grammar is *weakly* context free if its stringset is context free; a grammar is *strongly* context free if its treeset is also context free.

2.2 Applicability to Natural Language

Pullum and Gazdar (1982) survey the arguments up to the time they wrote for the non-contextfreeness of natural language. The most interesting were those that considered idealizations of linguistic phenomena in terms of the string duplicating language, ww. In each case they found the argument flawed: the phenomena in question did not yield languages whose stringsets were homomorphic to the duplication language. Bresnan et al. (1982) argue that Dutch is not strongly context free. Shieber (1985) provides a stringset argument about a dialect of Swiss-German, which has a class of verb phrases with cross-serial dependencies (through case marking) between NPs and their Vs, which establishes even the weak-non-context-freeness of natural language because of homomorphism to ww. Manaster-Ramer (1987) re-analyzes an argument considered by Pullum and Gazdar (1982) about Dutch and produces a

Hierarchy Level	Language Type	Model of Computation	Complexity
0	unrestricted phrase structure grammar (= r.e.)	Turing Machine (TM)	undecidable
1	context sensitive ($\subset$ recursive)	Linear Bounded Automata (LBA)	PSPACE
1.5	indexed	Nested Stack Automata (NSA)	NP-Complete
1.75	mildly context sensitive	Embeded Pushdown Automata (EPDA)	n^7
2	context free	Pushdown Automata (PDA)	n^3
3	regular	Finite State Machines (FSM)	linear

Table 1: Models of Grammar and Computation

corrected stringset argument that Dutch licences $a^n b^n c^n$ constructions, which are MCS. No known syntactic phenomenon requires greater than indexed language expressivity.

The point of this paper is to emphasize that although a particular Swiss-German dialect renders natural language syntax non-context free, it does not entail that natural languages, including the ones that license cross-serial dependencies, incur the worst case recognition complexity costs for indexed languages. In fact, we argue in the next section that ww is fairly straightforward to process. Essentially, we consider languages xx homomorphic to ww, where x can be either $\mathcal{L}3$ or $\mathcal{L}2$, and argue that the recognition for xx is no worse than worst case recognition for $\mathcal{L}3$ if $x \in \mathcal{L}3$ and no worse than the worst case for $\mathcal{L}2$ if $x \in \mathcal{L}2$, even though xx is itself indexed.

3 Cross-Serial Dependencies Are Not Hard to Process

It is always possible to compile less restrictive grammar formalisms into more restrictive covering formalisms, allowing different constituent analyses and potential stringset overgeneration. Metagrammatical techniques give an alternative that preserve coverage, but use special purpose processing. We suggest a parsing method for languages that rely on ww which does not cost a greater complexity fee than the worst case for parsing context free grammars. The method is metagrammatical and therefore akin to proposals put forward previously for handling coordination (Dahl and McCord, 1983) with logic grammars and TAGs (Shieber, 1995) or for extraposition (Milward, 1994). The method is constrained enough not to augment overall processing complexity, implying that ww does not require the worst case recognition complexity for its characteristic class, the MCS languages.

3.1 Why not?

Trivially, the string duplication languages can be recognized with time complexity proportional to the length of the string — if the string is of even length, and its first half is identical to the second half, then this can be established in just linear time. Though trivial in the sense of being about mere recognition, this is nonetheless interesting. In particular, under the reasonable hypothesis that humans are not in general reverse-wired[3] it is easier to process serial orders than their reverse. In this trivial recognition model we could take the serial ordering as primitive, but to use the same model as a recognizer for the *context free* string reversal languages would require an additional step of reversing the second half of the string before checking equivalence, which means the recognition complexity is $nlogn$. Thus, for trivial recognition the string duplication languages are easier to process than the string reversal languages. This is a concrete illustration that not every language costs the worst case recognition complexity for its expressivity class.

However, in the case of natural languages, parsing is of greater interest than mere recognition. A generalization of the recognizer method can be used inside a parsing approach as well. Suppose some i such that $i \geq 2$; suppose we want a recognizer for $\{ww | w \in \{a,b\}^*\}$ where $w \in \mathcal{L}i$, then we can use a parser that is no worse than cubic (if $i = 2$) and which can be linear (if $i = 3$) to determine if $w \in \mathcal{L}i$. Thus, if we parse exactly half of the string using a processor designed for languages in $\mathcal{L}i$, and then ascertain whether the remaining half is identical, then we remain in the

[3]While there actually is structural reverse wiring, psychological effects, like child learning of the distinction between left and right hands on themselves and on a person facing them, suggest that there is a difference in processing time required between recognizing a copy and an inverse copy. Another example comes from the recognition of rotated objects. There is a robust effect for which given a reference object and a rotated object-in-question it takes time linear in the amount of rotation to recognize the objects as copies. Mirror-image objects are isomorphic, yet it takes strictly more time to recognize reflected copies than to recognize nonreflected copies (Cooper, 1975).

same processing complexity class, since the identity check occurs after the parse and only requires linear time, but we also have structural information about the sentence as a whole. We know the structure of the first half of the string, and the second half of the string but not the structure of the second half (the grammar for w could be ambiguous), although we can assume that the second w was licensed by exactly the same tree structure as the first. This method also preserves a relative difference between parsing ww and ww^R, at least for $\mathcal{L}3$. Since ww^R can be represented directly within $\mathcal{L}2$ it can be argued that we should not be required to use the metagrammatical method of parsing it, just to keep symmetry with the duplication languages. Interestingly, if w is in $\mathcal{L}2$ and we use the metagrammatical parsing method, then ww^R also requires more processing time than ww for the same reason as the trivial case. Suppose instead that we allow ww^R to be parsed without using the metagrammatical method. In that case ww is relatively even easier to process since it costs $|w|^3$ to parse with the metagrammatical approach but ww^R will cost $(2|w|)^3$ in the direct approach. It might be claimed that just as we argue ww not to require the worst case complexity for its language class ($\mathcal{L}1.5$), neither need ww^R for $\mathcal{L}2$; but, the reversal language is a canonical example of a language that makes maximal use of the stack in the PDA. In any case, the metagrammatical method for parsing ww costs no more than just parsing strings in the characteristic language class of w.

If this were the complete story then we could only recognize languages homomorphic to the duplication languages. Clearly even the Zürich dialect of Swiss-German allows other constructions, all of which we can assume are context free (Pullum and Gazdar, 1982). Essentially we want to be able to write arbitrary $\mathcal{L}3$ or $\mathcal{L}2$ grammars and also be able to parse the string duplication language for whichever $\mathcal{L}i$ we choose. The language defined by such a union is no longer $\mathcal{L}i$, but will not contain arbitrary $\mathcal{L}1.5$ strings, and if $i = 3$ then the union will not even contain arbitrary context free strings. However, the situation is more involved than the basic approach since there needs to be a way to indicate where the metagrammatical approach is to be invoked. Add a single feature to the grammar interpreted by the processor as 'expect a copy'.[4]

1. $A \longrightarrow WB_{[c]}Y$

We allow context free productions of the form shown in (1), where A and B are nonterminals and W, Y are (possibly empty) sequences of terminals and nonterminals, B possibly occurring among the nonterminals of Y. For an ambiguous CFG, there is no guarantee that multiple instances of a nonterminal will rewrite to through the same sequence of productions to yield the same string.

There are any number of ways that this basic notation can be used in a metagrammatical approach. In the first instance, we take c to be a signal to the processor to generate an expectation for a duplicate of the terminal sequence that the nonterminal it is attached to gets rewritten to, and that this expectation must be satisfied by the next nonterminal of the same name and in the same local domain.[5] This approach will require that the sequence of terminals rewritten from the first B in (1) will be duplicated by the terminal sequence rewritten from the first instance of B (if any) that occurs in Y. The restriction will not hold of subsequent instances of the nonterminal marked for copying in the same local domain nor at different levels in the analysis. A stronger interpretation could require an expectation for the same constituent analysis of the nonterminal as well. Since we do not allow the feature to stack, the string-based method does not yield the full expressive power of indexed languages. The point is just that it's possible to keep a CF (or regular) grammar, and supplement the processor with a string-duplication operator which can be invoked at the subsentence level. This is sufficient to yield languages that more closely resemble the Zürich dialect in having other constructions besides the duplication construction, yet remaining efficiently processable.[6]

We have implemented the interpreter in a chart parser that can be used in either top-down or bottom-up fashion. Edges in the chart are marked with a category (some nonterminal or preterminal symbol from the grammar), constituents, substring span and expectations (along with a unique identifier for each edge). This is modified to include a list of constraints, which for the present purposes is presumed to be just duplication checks. An edge with no expectations is inactive (saturated) and one with expectations is active. In the completer step, when active edges combine with adjacent inactive edges whose category satisfies the current expectation of the active, the usual process of creating a new edge with one less expectation is augmented with another: if the current expectation has an associated copy feature, then the new edge is marked with a constraint interpreted by the parser as indicated above — the nonterminal symbol and the string spanned by the inactive edge are noted so

[4]Once we admit 'interpretability by the processor' we in principle have TM power. However we make quite restricted use of such interpretation. The rule format makes clear that it is less expressive than indexed grammars when interpreted directly.

[5]We take a local domain, in tree terms, as a node and the set of nodes that it immediately dominates.

[6]To get closer still to the Zürich dialect, we require that the duplication operator be applied at the level of preterminals, with complementation, to get the pairings of case-marked NPs and Vs.

that the next inactive edge of the same category (if one is expected) will have to span an identical string. Constraints of this form are not passed on after satisfied once, and are not passed out of the local domain. Within the same set of restrictions the implemented constraint could have been 'expect a reversed copy'. This would require computating the string's reverse before annotating the constraint list.

4 Discussion

The context free languages have already been studied from the perspective of minimal addition to incorporate copy languages. Savitch (1989) does exactly that by presenting the model of computation required for the class of languages defined by augmenting the CFLs with reduplication: a Reduplication PDA (RPDA). An RPDA is just a PDA which has a special type of symbol that can be put onto the stack to make the machine treat the part of the stack above it as if it were a queue. Essentially, this obtains the reversal behavior needed of a stack to process copy languages as well as reversals. Multiple instances of the special symbol can be placed on the stack. Savitch presents a characterization of the languages in terms of stringsets and the requisite computational structures. The family that we characterized above in terms of grammars are properly a subset of the languages recognized by RPDA, a restriction of RPDA languages which Savitch (1989) terms *simple RPDA languages*. The model of computation here is an RPDA in which only one special symbol is allowed on the stack at any one time. We have not proven the equivalence we conjecture between our metagrammatical method and the *reduplication context-free grammars* (RCFGs) that Savitch introduces as generative of simple RPDA languages. Savitch's (1989) grammars are stated in terms of rule schemata (a finite set) that generate potentially infinite sets of rewrite rules. This is the tradeoff between doing things metagrammatically and directly.

Joshi and Rambow (Joshi, 1990; Rambow and Joshi, 1994) have also considered the performance data associated with processing crossed vs. nested dependencies and present an alternative computation model, the *bottom-up embedded PDA* (BEPDA), designed for a variant of tree-adjoining grammar (it uses a stack of stacks and a more complex operation for emptying the stack). Rambow and Joshi (1994) use the processing model to demonstrate that it can account for the difference between crossed and nested dependencies in terms of the amount of time associated objects spend in the pushdown store of the BEPDA using a mildly context free language model that captures dependencies directly, rather than metagrammatically.[7]

[7]Joshi (1990) gives a similar analysis for EDPAs.

Essentially, their analysis concludes the same: when judging string isomorphisms, it is easier to make the judgment of identically ordered pairs than it is to reversely ordered pairs. Thus, the cross-serial dependencies needn't cost the worst case complexity for parsing indexed or mildly context sensitive languages. Parsing ww languages requires, at worst, the worst case complexity of parsing w in whichever language class w is restricted to. Shieber (1985) pointed out without proof that the nonCF data associated Zürich dialect is linearly parsable; our task has been to clarify how this follows from the language theory.

4.1 A Caveat

For efficient processing of ww to entail corresponding complexity for natural languages that license cross-serial dependencies hinges crucially on there being efficiently computable homomorphisms between the natural language and the string duplication languages. This is an open question. However, given that empirical work that compares processing of crossed and nested dependencies and concludes that the cross-serial dependencies are preferred to nested ones (Bach et al., 1986), and given our argument that cross-serial dependencies are in theory easier to process, we feel it reasonable to entertain the assumption that something such exists. This does not require us to assume that people actually use context-free grammars and compute homomorphisms in order to understand natural languages, just that the computational model should be at least approximately as efficient as people.

4.2 Implications

Our metagrammatical approach to dealing with cross serial dependencies involves the assumption of an operation for testing string duplication. We hinted earlier that we feel there to be sufficient reason to believe that copy-checking is a basic cognitive function, and although we don't suppose that people have built in production systems and processors isomorphic to our chart parser and base language, we do think that this copy-checking is invoked in the processing of crossed dependencies. Our approach to accounting for the processing complexity that the string duplication languages should take does make empirical predictions and these can be tested. For instance, if it is the case that such a mechanism exists, then patterns of string-copy disfluency should occur with different frequency in languages that license cross-serial dependencies than in those that do not. A string-copy disfluency is just one that involves a repeat of part of the sentence uttered so far:

1. *We went to the to the store to buy some flour.*

The idea is that speakers of languages with ww homomorphisms have a different pattern of invoking copy-checking than those who speak lan-

guages that do not admit cross serial dependencies. These differences should be manifest in speech corpora like those that are currently being accumulated (Anderson et al., 1992; Miller, 1995), but which need augmentation by a corpus derived from copy-language dialects. Verifying this would, for example, establish whether the copied strings need to be constituents, and this has a bearing on whether processing models designed for incremental interpretation (Milward, 1992) are the best descriptors of human performance.[8] We do not offer arguments that our metagrammatical approach is the best description of human processing of cross-serial dependencies, just that it is another theoretical justification for the difference in processing nested dependencies and efficient processing of crossed dependencies.

Acknowledgements

Vogel is grateful to the SFB 340 for funding his stay Stuttgart; Hahn acknowledges the support of ESRC grant No. R004293341442; Branigan, EPSRC research studentship No. 92315069. All would like to thank Catherine Collin, Tomaž Erjavec, Tsutomu Fujinami, Merce Prat, Fred Popowich, Mark Steedman, and the anonymous reviewers.

References

Alfred V. Aho. 1968. Indexed grammars—an extention to context-free grammars. *Journal of the Association for Computing Machinery*, 15(4):647–671.

Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth H. Boyle, Gwyneth M. Doherty, Simon C. Garrod, Stephen D. Isard, Jacqueline C. Kowtko, Jan M. McAllister, Jim Miller, Catherine F. Sotillo, Henry S. Thompson, and Regina Weinert. 1992. The HCRC Map Task corpus. *Language and Speech*, 34(4):351–366.

Emmon Bach, Colin Brown, and William Marslen-Wilson. 1986. Cross and nested dependencies in german and dutch: A psycholinguistic study. *Language and Cognitive Processes*, 1(4):249–262.

Joan Bresnan, Ron Kaplan, Stanley Peters, and Annie Zaenen. 1982. Cross-serial dependencies in dutch. *Linguistic Inquiry*, 13(4):613–35.

Lynn Cooper. 1975. Mental rotation of random two-dimensional shapes. *Cognitive Psychology*, 7:23–43.

Veronica Dahl and Michael McCord. 1983. Treating coordination in logic grammars. *American Journal of Computational Linguistics*, 9(2):69–91.

[8]Note that the English 'respectively' constructions require a special intonational behavior in the sing-song litany-voice that is required for a speaker to make an extended 'respectively' construction interpretable, thus arguments for specifically metagrammatical treatment do exist (where intonational facts are considered evidence for a signal to the processor to do something unusual).

Gerald Gazdar and Geoffrey Pullum. 1985. Computationally relevant properties of natural languages and their grammars. Technical Report CSLI-85-24, Stanford: Center for the Study of Language and Information.

Gerald Gazdar. 1985. Applicability of indexed grammars to natural language. Technical Report CSLI-85-34, Stanford: Center for the Study of Language and Information.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Co., Reading MA.

Aravind K. Joshi, K. Vijay-Shanker, and David Weir. 1989. The convergence of mildly context-sensitive grammar formalisms. Technical Report MS-CIS-89-14; LINC LAB 144, Department of Computer and Information Science University of Pennsylvania, Philadelphia, PA.

Aravind Joshi. 1990. Processing crossed and nested dependencies: An automaton perspective on the psycholinguistic results. *Language and Cognitive Processes*, 5(1):1–27.

Alexis Manaster-Ramer. 1987. Dutch as a formal language. *Linguistics and Philosophy*, 10(2):221–46.

Jim Miller. 1995. Focus in the languages of europe. To appear in G. Bernini (ed.) *Pragmatic organization in the languages* (Volume I. of *Typology of the languages of Europe*). Mouton-de Gruyter.

David Milward. 1992. Dynamics, dependency grammar and incremental interpretation. In *COLING92*, pages 1095–9.

David Milward. 1994. Dynamic dependency grammar. *Linguistics and Philosophy*, 17:561–605.

Geoffrey Pullum and Gerald Gazdar. 1982. Natural languages and context-free languages. *Linguistics and Philosophy*, 4:471–504.

Owen Rambow and Aravind Joshi. 1994. A processing model for free word order languages. In L. Frazier C. Clifton, Jr. and K. Rayner, editors, *Perspectives on Sentence Processing.* Lawrence Erlbaum.

Walter Savitch. 1989. A formal model for context-free languages augmented with reduplication. *Computational Linguistics*, 15(4):250–61.

Stuart Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–43.

Stuart Shieber. 1995. What is wrong with tags. Invited talk at the *Seventh Conference of the European Chapter of the Association for Computational Linguistics.* Belfield, Dublin, Ireland.

Edward P. Stabler. 1994. The finite connectivity of linguistic structure. In Lyn Frazier Charles Clifton, Jr. and Keith Rayner, editors, *Perspectives on Sentence Processing.* Hillsdale, NJ: Lawrence Erlbaum.

Carl Vogel and Tomaž Erjavec. 1994. Restricted discontinuous phrase structure grammar and its ramifications. In Carlos Martin-Vide, editor, *Current Issues in Mathematical Linguistics.* The Netherlands: Elsevier Science Publishers.