PREDICTIVE COMBINATORS: A METHOD FOR EFFICIENT PROCESSING OF COMBINATORY CATEGORIAL GRAMMARS

Kent Wittenburg
MCC, Human Interface Program
3500 West Balcones Center Drive
Austin, TX 78759
&
Department of Linguistics
University of Texas at Austin
Austin, TX 78712

ABSTRACT

Steedman (1985, 1987) and others have proposed that Categorial Grammar, a theory of syntax in which grammatical categories are viewed as functions, be augmented with operators such as functional composition and type raising in order to analyze "noncanonical" syntactic constructions such as wh- extraction and node raising. A consequence of these augmentations is an explosion of semantically equivalent derivations admitted by the grammar. The present work proposes a method for circumventing this spurious ambiguity problem. It involves deriving new, specialized combinators and replacing the orginal basic combinators with these derived ones. In this paper, examples of these *predictive combinators* are offered and their effects illustrated. An algorithm for deriving them, as well as a discussion of their semantics, will be presented in forthcoming work.

Introduction

In recent years there has been a resurgence of interest in Categorial Grammar (Adjukeiwicz 1935; Bar-Hillel 1953). The work of Steedman (1985, 1987) and Dowty (1987) is representative of one recent direction in which Categorial Grammar (CG) has been taken, in which the operations of functional composition and type raising have figured in analyses of "noncanonical" structures such as wh- dependencies and nonconstituent conjunction. Based on the fact that such operations have their roots in the *combinatory calculus* (Curry and Feys 1958), this line of Categorial Grammar has come to be known as Combinatory Categorial Grammar (CCG). While such an approach to syntax has been demonstrated to be suitable for computer implementation with unification-based grammar formalisms (Wittenburg 1986a), doubts have arisen over the efficiency with which such grammars can be processed. Karttunen (1986), for instance, argues for an alternative to rules of functional composition and type raising in CGs on such grounds.[1] Other researchers working with Categorial Unification Grammars consider the question of what method to use for long-distance dependencies an open one (Uszkoreit 1986; Zeevat, Klein, and Calder 1986).

The property of Combinatory Categorial Grammars that has occasioned concerns about processing is spurious ambiguity: CCGs that directly use functional composition and type raising admit alternative derivations that nevertheless result in fully equivalent parses from a semantic point of view. In fact, the numbers of such semantically equivalent derivations can multiply at an alarming rate. It was shown in Wittenburg (1986a) that even constrained versions of functional composition and type raising can independently cause the number of semantically equivalent derivations to grow at rates exponential in the length of the input string.[2] While this spurious ambiguity property may not seem to be a particular problem if a depth-first (or best-first) parsing algorithm is used--after all, if one can get by with producing just one derivation, one has no reason to go on generating the remaining equivalent ones--the fact is that both in cases where the parser ultimately fails to generate a derivation and where one needs to be prepared to generate all and only genuinely (semantically) ambiguous parses, spurious ambiguity may be a roadblock to efficient parsing of natural language from a practical perspective.

The proposal in the present work is aimed toward eliminating spurious ambiguity from the form of Combinatory Categorial Grammars that are actually used during parsing. It involves deriving a new set of combinators, termed predictive combinators, that replace the basic forms of functional composition and type raising in the original grammar. After first reviewing the theory of Combinatory Categorial Grammar and the attendant spurious ambiguity problem, we proceed to the subject of these derived combinators. At the conclusion, we compare this approach to other proposals.

---

[1] Karttunen suggests that these operations, at least in their most general form, are computationally intractable. However, it should be noted that neither Steedman nor Dowty has suggested that a fully general form of type raising, in particular, should be included as a productive rule of the syntax. And, as Friedman, Dai, and Wang (1986) have shown, certain constrained forms of these grammars that nevertheless include functional composition are weakly context-free. Aravind Joshi (personal communication) strongly suspects that the generative capacity of the grammars that Steedman assumes, say, for Dutch, is in the same class with Tree Adjoining Grammars (Joshi 1985) and Head Grammars (Pollard 1984). Thus, computational tractability is, I believe, not at issue for the particular CCGs assumed here.

[2] The result in the case of functional composition was tied to the Catalan series (Knuth 1975), which Martin, Church and Patil (1981) refer to as "almost exponential". For a particular implementation of type raising, it was 2^{n-1}. The fact that derivations grow at such a rate, incidentally, does not mean that these grammars, if they are weakly context-free, are not parsable in n^3 time. But it is such ambiguities that can occasion the worst case for such algorithms. See Martin, Church, and Patil (1981) for discussion.

Overview of CCG

The theory of Combinatorial Categorial Grammar has two main components: a categorial lexicon that assigns grammatical categories to string elements and a set of combinatory rules that operate over these categories.[3]

Categorial lexicon

The grammatical categories assigned to string elements in a Categorial Grammar can be basic, as in the category CN, which might be assigned to the common noun *man*, or they may be of a more complex sort, namely, one of the so-called *functor* categories. Functor categories are of the form X|Y, which is viewed as a function from categories of type Y to categories of type X. Thus, for instance, a determiner such as *the* might be assigned the category NP|CN, an indication that it is a function from common nouns to noun phrases. An example of a slightly more complex functor category would be tensed transitive verbs, which might carry the category (S|NP)|NP. This can be viewed as a second order function from (object) noun phrases to another function, namely S|NP, which is itself a function from (subject) noun phrases to sentences.[4] (Following Steedman, we will sometimes abbreviate this finite verb phrase category as the symbol FVP.) Directionality is indicated in the categories with the following convention: a right-slanting slash indicates that the argument Y must appear to the right of the functor, as in X/Y; a left-slanting slash indicates that the argument Y must appear to the left, as in X\Y.[5] A vertical slash in this paper is to be interpreted as specifying a directionality of *either* left or right.

Combinatorial rules

Imposing directionality on categories entails including two versions of the basic functional application rule in the grammar. Forward functional application, which we will note as 'fa>', is shown in (1a), backward functional application ('fa<') in (1b).

(1)

a. Forward Functional Application (fa>)

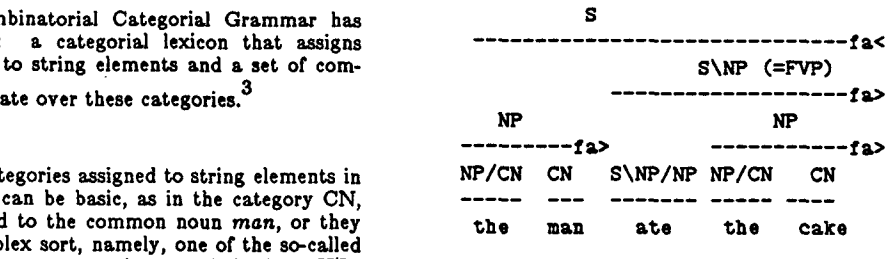
X/Y Y => X

b. Backward Functional Application (fa<)

Y X\Y => X

An example derivation of a canonical sentence using just these combinatory rules is shown in (2).

(2)



Using just functional application results in derivations that typically mirror traditional constituent structure. However, the theory of Combinatory Categorial Grammar departs from other forms of Categorial Grammar and related theories such as HPSG (Pollard 1985; Sag 1987) in the use of functional composition and type raising in the syntax, which occasions partial constituents within derivations. Functional composition is a combinatory operation whose input is two functors and whose output is also a functor composed out of the two inputs. In (3) we see one instance of functional composition (perhaps the only one) that is necessary in English.[6]

(3) Forward functional composition (fc>)

X/Y Y/Z => X/Z

The effect of type raising, which is to be taken as a rule schema that is instantiated through individual unary rules, is to change a category that serves as an argument for some functor into a particular kind of complex functor that takes the original functor as its new argument. An instance of a type-raising rule for topicalized NPs is shown in (4a)[7]; a rule for type-raising subjects is shown in (4b) in two equivalent notations.

(4)

a. Topicalization (top)

NP => S/(S/NP)

b. Subject type-raising (str)

NP => S/FVP
[NP => S/(S\NP)]

The rules in (3) and (4) can be exploited to account for unbounded dependencies in English. An instance of topicalization is shown in (5).

---

[3] In Wittenburg (1986a), a set of unary rules is also assumed that may permute arguments and shift categories in various ways, but these rules are not germane to the present discussion.

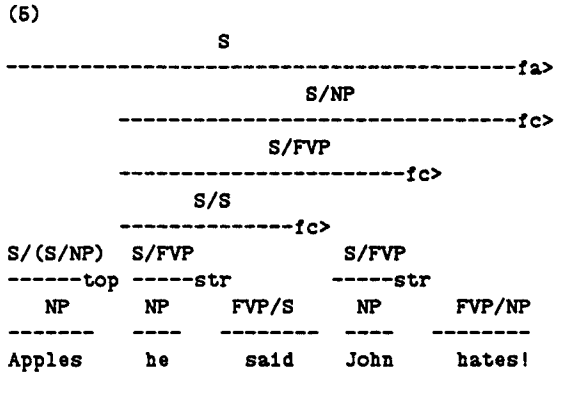
[4] When parentheses are omitted from categories, the bracketing is left-associative, i.e., S|NP|NP receives exactly the same interpretation as (S|NP)|NP.

[5] Note that X is the range of the functor in both these expressions and Y the domain. This convention does not hold across all the categorial grammar literature.

[6] Functional composition is known as B in the combinatory calculus (Curry and Feys 1958).

[7] The direction of the slash in the argument category poses an obvious problem for cases of subject extraction, a topic which we will not have space to discuss here. But see Steedman (1987).

(5)

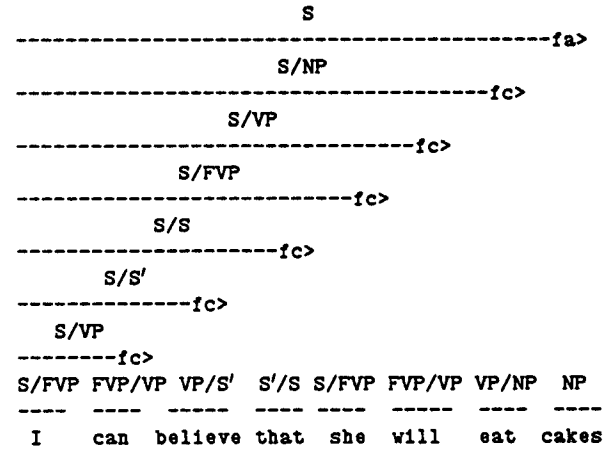


Such analyses of unbounded dependencies get by without positing special conventions for percolating slash features, without empty categories and associated ϵ-rules, and without any significant complications to the string-rewriting mechanisms such as transformations. The two essential ingredients, namely, type-raising and functional composition, are operations of wide generality that are sufficient for handling node-raising (Steedman 1985; 1987) and other forms of nonconstituent conjunction (Dowty 1987). Using these methods to capture unbounded dependencies also preserves a key property of grammars, namely, what Steedman (1985) refers to as the adjacency property, maintained when string rewriting operations are confined to concatenation. Grammars which preserve the adjacency property, even though they may or may not be weakly context-free, nevertheless can make use of many of the parsing techniques that have been developed for context-free grammars since the applicability conditions for string-rewriting rules are exactly the same.

The spurious ambiguity problem

A negative consequence of parsing directly with the rules above is an explosion in possible derivations. While functional composition is required for long-distance dependencies, i.e., a CCG without such rules could not find a successful parse, they are essentially optional in other cases. Consider the derivation in (6) from Steedman (1985).[8]

(6)



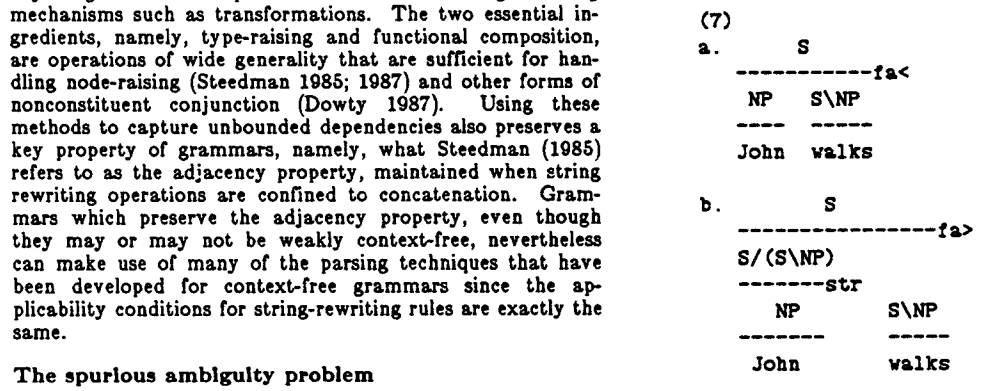
[8]We do not show the subject type-raising rules in this derivation, but assume they have already applied to the subject NPs.

This is only one of many well-formed derivations for this sentence in the grammar. The maximal use of functional composition rules gives a completely left branching structure to the derivation tree in (6); the use of only functional application would give a maximally right-branching structure; a total of 469 distinct derivations are in fact given by the grammar for this sentence.

Given that derivations using functional composition can branch in either direction, spurious ambiguity can arise even in sentences which depend on functional composition. Note, for instance, that if we topicalized *cakes* in (6), we would still be able to create the partial constituent S/NP bridging the string *I can believe that she will eat* in 132 different ways.

Some type-raising rules can also provoke spurious ambiguity, leading in certain cases to an exponential growth of derivations in the length of the string (Wittenburg 1986a). Here again the problem stems from the fact that type-raising rules can apply not just in cases where they are needed, but also in cases where derivations are possible without type raising. An example of two equivalent derivations made possible with subject type-raising is shown in (7).

(7)



Note that spurious ambiguity is different from the classic ambiguity problem in parsing, in which differing analyses will be associated with different attachments or other linguistically significant labelings and thus will yield differing semantic results. It is a crucial property of the ambiguity just mentioned that there is no difference with respect to their fully reduced semantics. While each of the derivations differs from the others in the presence or absence of some intermediate constituent(s), the semantics of the rules of functional composition and type raising ensure that after full reductions, the semantics will be the same in every case.[9]

Predictive combinators

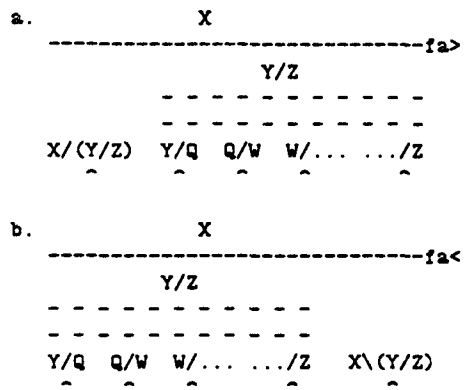
Here we show how it is possible to eliminate spurious ambiguity while retaining the the analyses (but not the derivations) of long-distance dependencies just shown. The proposal involves deriving new combinatory rules that replace functional composition and the ambiguity-producing type-raising rules in the grammar. The difference between the original grammar and this derived one is that the new combinators will by nature be restricted to just those derivational contexts where they are necessary whereas in the original grammar, these rules can apply in a wide range of contexts.

The key observation is the following. Functional composition and certain type raising rules are only necessary (in the sense that a derivation cannot be had without them) if

[9]This equivalence holds also if the "semantics" consists of intermediate f-structures built by means of graph-unification-based formalisms as in Wittenburg (1986a).

categories of the form X|(Y|Z) appear at one end of derivational substring. This category type is distinguished by having an argument term that is itself a functor. As proved by Dowty (1987), adding functional composition to Categorial Grammars that admit no categories of this type has no effect on the set of strings these grammars can generate, although of course it does have an effect on the number of derivations allowed. When CGs do allow categories of this type, then functional composition (and some instances of type raising) can be the crucial ingredient for success in derivations like those shown in schematic form in (8).

(8)



These schemata are to be interpreted as follows. The category strings shown at the bottom of (8a) and (8b) are either lexical category assignments OR (as indicated by the carets) categories derivable in the grammar with rules of functional application or unary rules such as topicalization. Recall that CCGs with such rules alone have no spurious ambiguity problem. The category strings underneath the wider dashed lines are then reducible via (type raising and) functional composition into functional arguments of the appropriate sort that are only then reduced via functional application to the X terms.[10] It is this part of the derivation, i.e., the part represented by the pair of wider dashed lines, in which spurious ambiguity shows up. Note that (5) is intended to be an example of the sort of derivation being schematized in (8a): the topicalization rule applies underneath the leftmost category to produce the X/(Y/Z) type; all other categories in the bottommost string in (8a) correspond to lexical category assignments in (5).

There are two conditions necessary for eliminating spurious ambiguity in the circumstances we have just laid out. First, we must make sure that function composition (and unary rules like subject type-raising) only apply when a higher type functor appears in a substring, as in (8). When no such higher type functors appears, the rules must then be absent from the picture--they are unnecessary. Second, we must be sure that when function composition and unary rules like subject type-raising do become involved, they produce unique derivations under conditions like (8), avoiding the spurious ambiguity that characterizes function composition and type raising as they have been stated earlier.

[10]While we have implied (as evidenced by the right-leaning slashes on intermediate categories) that forward functional composition is the relevant composition rule, backwards functional composition could also be involved in the reduction of substrings, as could type raising.

The natural solution for enforcing the first condition is to involve categories of type X|(Y|Z) in the derivations from the start. In other words, restricting the application of functional composition and the relevant type-raising rules is possible if we can incorporate some sort of top-down, or predictive, information from the presence of categories of type X|(Y|Z). Standard dotted rule techniques found in Earley deduction (Earley 1970) and active chart parsing (Kay 1980) offer one avenue with which to explore the possibility of adding such control information to a parser. However, since the information carried by dotted rules in algorithms designed for context-free grammars has a direct correlate in the slashes already found in the categories of a Categorial Grammar, we can incorporate such predictive information into our grammar in categorial terms. Specifically, we can derive new combinatorial rules that directly incorporate the "top-down" information. I call these derived combinatorial rules *predictive combinators*.[11]

It so happens that these same predictive combinators will also enforce the second condition mentioned above, by virtue of the fact that they are designed to branch uniformly from the site of the higher type functor to the site of the "gap". For cases of leftward extraction (8a), derivations will be uniformly left-branching. For cases of rightward extraction (8b), derivations will be uniformly right-branching. It is our conjecture that CCGs can be compiled so as to force uniform branching in just this way without affecting the language generated by the grammar and without altering the semantic interpretations of the results. We will now turn to some examples of the derived combinatory rules in order to see how they might produce such derivations.

The first predictive combinator we will consider is derived from categories of type X/(Y/Z) and forward functional composition of the argument term of this category. It is designed for use in category strings like those that appear in (8a). The new rule, which we will call *forward-predictive functional composition*, is shown in (9).

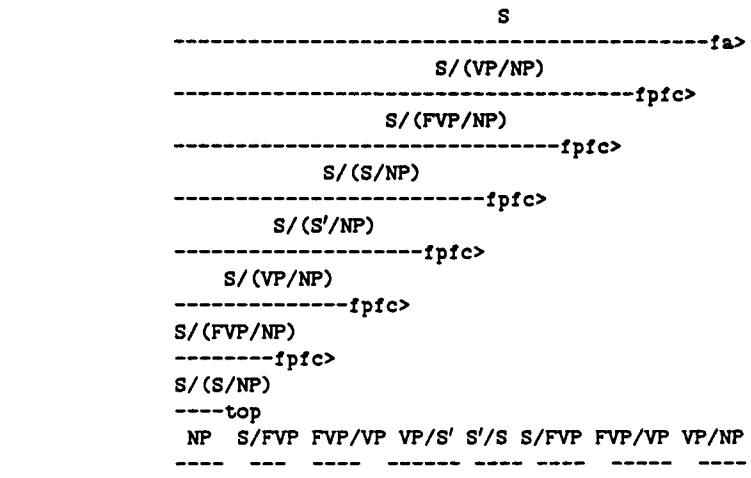
(9) Forward-predictive forward functional composition (fpfc>)

X/(Y/Z) Y/W => X/(W/Z)

Assuming a CCG with the rule in (9) in place of forward functional composition, we are able to produce derivations such as (10). Here, as in some earlier examples, we assume Subject type-raising has already applied to subject NP categories.

[11]There is a loose analogy between these predictive combinators and the concept of *supercombinators* first proposed by Hughes (1982). Hughes proposed, in the context of compilation techniques for applicative programming languages, methods for deriving new combinators from actual programs. He used the term *supercombinators* to distinguish this derived set from the fixed set of combinators proposed by Turner (1979). By analogy, predictive combinators in CCGs are derived from actual categories and rules defined in specific Combinatory Categorial Grammars. There are in principle infinitely many of them, depending on the particulars of individual grammars, and thus they can be distinguished from the fixed set of "basic" combinatorial rules for CCGs proposed by Steedman and others.

(10)



We took note above of the fact that there were at least 132 distinct derivations for the sentence now appearing in (10) with CCGs using forward functional composition directly. With forward-predictive forward functional composition in its place, there is one and only one derivation admitted by the grammar, namely, the one shown. In order to see this, note that the string to the right of *cakes* is irreducible with any rules now in the grammar. Only fpfc> can be used to reduce the category string, and it operates in a necessarily left branching fashion, triggered by an X/(Y/Z) category at the left end of the string.

A second predictive combinator necessary to fully incorporate the effects of forward functional composition is a version of predictive functional composition that works in the reverse direction, i.e., *backward-predictive forward functional composition*. It is necessary for category strings like those in (8b), which are found in CCG analyses of English node raising (Steedman 1985). The rule is shown in (11).

(11) Backward-predictive forward functional composition (bpfc>)

W/Z X\(Y/Z) => X\(Y/W)

Intuitively, the difference between the backward-predictive and the forward-predictive versions of function composition is that the forward version passes the "gap" term rightward in a left-branching subderivation, whereas the backward version passes the "principle functor" in the argument term leftward in a right-branching subderivation. We see an example of both these rules working in the case of right-node-raising shown in (12). It is assumed here, as in Steedman (1985), that the conjunction category involves finding like bindings for category variables corresponding to each of the conjuncts. We use A and B below as names for these variables, and the vertical slash must be interpreted here as a directional variable as well. Note that bindings of variables in rule applications, as say the X term in the instance of pbfc>, can involve complex parenthesized categories (recall that we assume left-association) in addition to basic ones.

(12)

It is our current conjecture that replacing forward functional composition in CCGs with the two rules shown will eliminate any spurious ambiguity that arises directly from this composition rule. However, we have yet to show how spurious ambiguity from subject type-raising can be eliminated. The strategy will be the same, namely, to replace subject type-raising with a set of predictive combinators that force uniformly branching subderivations in cases requiring function composition.

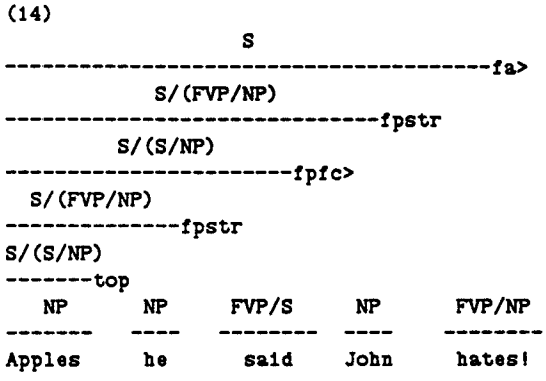
For compiling out unary rules generally, it is necessary to consider all existing combinatory rules in the grammar. In our current example grammar, we have four rules to consider in the compilation process: forward and backward (predictive) functional application, and the newly derived predictive function composition rules as well. Subject type-raising can in fact be merged with each of the four combinatory rules mentioned to produce four new predictive combinators, each of which have motivation for certain cases of node-raising. Here we will look at just one example, namely, the rule necessary to get leftward "movement" (topicalization and wh- extraction) over subjects. Such a rule can be derived by merging subject type-raising with the right daughter of the new forward-predictive forward function composition rule, maintaining all bindings of variables in the process. This new rule which, in the interest of brevity, we call *forward-predictive subject type raising* is shown in (13).

(13) Forward-predictive subject type raising (fpstr)

X/(S/Z) NP => X/(FVP/Z)

The replacement of subject type raising with the predictive combinator in (13) eliminates spurious derivations such as (7b). Instead, the effects of subject type raising will only be realized in derivations such as (14), which are marked by requiring the effects of subject type raising to get a derivation at all.

(14)



The predictive combinator rules in (9), (11), and (13) are examples of a larger set necessary to completely eliminate spurious ambiguity from most Combinatory Categorial Grammars. In the class of function composition rules, we have considered only forward functional composition in this paper, but many published CCG analyses assume rules of backward functional composition as well. As we mentioned, compiling out type-raising rules may involve adding as many new combinators as there are general combinatory rules in the grammar previously. Other unary rules that produce spurious ambiguity may require even more predictive combinators. The rule of subject-introduction proposed in Wittenburg (1986a) may be one such example.

There are of course costs involved in increasing the size of a rule base by enlarging the grammar through the addition of predictive combinators. However, the size of a rule base is well known to be a constant factor in asymptotic analyses of parsing complexity (and the rule base for Categorial Grammars is very small to begin with anyway). On the other hand, the cost of producing spuriously ambiguous derivations with grammars that include functional composition is at least polynomial for the best known parsing algorithms. The reasoning is as follows. Based on the (optimistic) assumption that relevant CCGs are weakly context-free, they are amenable to parsing in n^3 time by, say, the Earley algorithm (Earley 1970).[12] As alluded to earlier in footnote 2, "all-ways ambiguous" grammars, a characterization that holds for CCGs that use function composition directly, occasion the worst case for the Earley algorithm, namely n^3. This is because all possible well-formed bracketings of a string are in fact admitted by the grammar in these worst cases (as exemplified by (6)) and the best the Earley algorithm can do when filling out a chart (or its equivalent) in such circumstances is $O(n^3)$. The methods presented here for normalizing CCGs through predictive combinators eliminate this particular source of worst case ambiguity. Asymptotic parsing complexity will then be no better or worse than the grammar and parser yield independently from the spurious ambiguity problem. Further, whatever the worst case results are, there will presumably be statistically fewer instances of the worst cases since an omnipresent source of all-ways ambiguity will have been eliminated.

Work on predictive combinators at MCC is ongoing. At the time of this writing, an experimental algorithm for compiling a predictive form of CCGs, given a base form along the lines of Steedman (1985), has been implemented for CCGs expressed in a PATR-like unification grammar formalism (Shieber 1984). We believe from experience that our algorithm is correct and complete, although we do not have a formal proof at this point. A full formal characterization of the problem, along with algorithms and accompanying correctness proofs, is forthcoming.

Comparison with previous work

Previous suggestions in the literature for coping with spurious ambiguity in CCGs are characterized not by eliminating such ambiguity from the grammar but rather by attempting to minimize its effects during parsing.[13] Karttunen (1986) has suggested using equivalence tests during processing; in his modified Earley chart parsing algorithm, a subconstituent is not added to the chart without first testing to see if an equivalent constituent has already been built.[14] In its effects on complexity, this check is really no different than a step already present in the Earley algorithm: an Earley state (edge) is not added to a state set (vertex) without first checking to see if it is a duplicate of one already there.[15] The recognition algorithm does nothing with duplicates; for the Earley parsing algorithm, duplicates engender an additional small step involving the placement of a pointer so that the analysis trees can be recovered later. Duplicates generated from functional composition (or from other spurious ambiguity sources) require a treatment no different than Earley's duplicates except that no pointers need to be added in parsing--their derivations are simply redundant from a semantic point of view and thus they can be ignored for later processing. Karttunen's proposal does not change the worst-case complexity results for Earley's algorithm used with CCGs as discussed above and thus does not offer much relief from the spurious ambiguity problem. However, parsing algorithms such as Karttunen's that check for duplicates are of course superior from the point of view of asymptotic complexity to parsing algorithms which fail to make checks. The latter sort will on the face of it be exponential when faced with ambiguity as in (6) since each of the independent derivations corresponding to the Catalan series will have to be enumerated independently.

In earlier work (Wittenburg 1986a, 1986b), I have suggested that heuristics used with a best-first parsing algorithm can help cope with spurious ambiguity. It is clear to me now that, while more intelligent methods for directing the search can significantly improve performance in the average case, they should not be viewed as a solution to spurious ambiguity in general. Genuine ambiguity and unparsable input in natural language can force the parser to search exhaustively with respect to the grammar. While heuristics used even with a large search space can provide the means for tuning performance for the "best" analyses, the search space itself will determine the results in the "worst" cases. Compiling the grammar into a normal form based on the notion of predictive combinators makes exhaustive search more palatable, whatever the enumeration order, since the search

---

[12] Even if the CCGs in question are not weakly context-free, it is still likely that asymptotic complexity results will be polynomial unless the relevant class is not within that of the limited extensions to context-free grammars that include Head Grammars (Pollard 1984) and TAGs (Joshi 1985). Pollard (1984) has a result of n^7 for Head Grammars.

[13] This characterization also apparently holds for the proposals from Pareschi and Steedman (1987) being presented at this conference.

[14] While Karttunen's categorial fragment for Finnish does not make direct use of functional composition and type raising, it nevertheless suffers from spurious ambiguity of a similar sort stemming from the nature of the categories and functional application rules he defines.

[15] The n^3 result crucially depends on this check, in fact.

space itself is vastly reduced. Heuristics (along with best-first methods generally) may still be valuable in the reduced space, but any enumeration order will do. Thus Earley parsing, best-first enumeration, and even LR techniques are still all consistent with the proposal in the current work.

ACKNOWLEDGEMENTS

The research on which this paper is based was carried out in connection with the Lingo Natural Language Interface Project at MCC. I am grateful to Jim Barnett, Elaine Rich, Greg Whittemore, and Dave Wroblewski for discussions and comments. This work has also benefitted from discussions with Scott Danforth and Aravind Joshi, and particularly from the helpful comments of Mark Steedman.

REFERENCES

Adjukiewicz, K. 1935. Die Syntaktische Konnexitat. Studia Philosophica 1:1-27. [English translation in Storrs McCall (ed.). Polish Logic 1920-1939, pp. 207-231. Oxford University Press.]

Bar-Hillel, Y. 1953. A Quasi-Arithmetical Notation for Syntactic Description. Language 29: 47-58. [Reprinted in Y. Bar-Hillel, Language and Information, Reading, Mass.: Addison-Wesley, 1964, pp. 61-74.]

Curry, H., and R. Feys. 1958. Combinatory Logic: Volume 1. Amsterdam: North Holland.

Dowty, D. 1987. Type Raising, Functional Composition, and Non-Constituent Conjunction. To appear in R. Oehrle, E. Bach, and D. Wheeler (eds.), Categorial Grammars and Natural Language Structures, Dordrecht.

Earley, J. 1970. An Efficient Context-Free Parsing Algorithm. Communications of the ACM 13:94-102.

Friedman, J., D. Dai, and W. Wang. 1986. The Weak Generative Capacity of Parenthesis-Free Categorial Grammars. Technical report no. 86-001, Computer Science Department, Boston University, Boston, Massachusetts.

Hughes, R. 1982. Super-combinators: a New Implementation Method for Applicative Languages. In Symposium on Lisp and Functional Programming, pp. 1-10, ACM.

Joshi, A. 1985. Tree Adjoining Grammars: How Much Context-Sensitivity is Required to Provide Reasonable Structural Structural Descriptions? In D. Dowty, L. Karttunen, and A. Zwicky (eds.), Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives. Cambridge University Press.

Karttunen, L. 1986. Radical Lexicalism. Paper presented at the Conference on Alternative Conceptions of Phrase Structure, July 1986, New York.

Kay, M. 1980. Algorithm Schemata and Data Structures in Syntactic Processing. Xerox Palo Alto Research Center, tech report no. CSL-80-12.

Knuth, D. 1975. The Art of Computer Programming. Vol. 1: Fundamental Algorithms. Addison Wesley.

Martin, W., K. Church, and R. Patil. 1981. Preliminary Analysis of a Breadth-First Parsing Algorithm: Theoretical and Experimental Results. MIT tech report no. MIT/LCS/TR-261.

Pareschi, R., and M. Steedman. 1987. A Lazy Way to Chart Parse with Categorial Grammars, this volume.

Pollard, C. 1984. Generalized Phrase Structure Grammars, Head Grammars, and Natural Languages. Ph.D. dissertation, Stanford University.

Pollard, C. 1985. Lecture Notes on Head-Driven Phrase Structure Grammar. Center for the Study of Language and Information, Stanford University, Palo Alto, Calif.

Sag, I. 1987. Grammatical Hierarchy and Linear Precedence. To appear in Syntax and Semantics, Volume 20: Discontinuous Constituencies, Academic.

Shieber, S. 1984. The Design of a Computer Language for Linguistic Information. Proceedings of Coling84, pp. 362-366. Association for Computational Linguistics.

Steedman, M. 1985. Dependency and Coordination in the Grammar of Dutch and English. Language 61:523-568.

Steedman, M. 1987. Combinators and Grammars. To appear in R. Oehrle, E. Bach, and D. Wheeler (eds.), Categorial Grammars and Natural Language Structures, Dordrecht.

Turner, D. 1979. A New Implementation Technique for Applicative Languages. Software -- Practice and Experience 9:31-49.

Uszkoreit, H. 1986. Categorial Unification Grammars. In Proceedings of Coling 1986, pp. 187-194.

Wittenburg, K. 1986a. Natural Language Parsing with Combinatory Categorial Grammars in a Graph-Unification-Based Formalism. Ph.D. dissertation, University of Texas at Austin.[Some of this material is available through MCC tech reports HI-012-86, HI-075-86, and HI-179-86.]

Wittenburg, K. 1986b. A Parser for Portable NL Interfaces using Graph-Unification-Based Grammars.

In Proceedings of AAAI-86, pp. 1053-1058.

Zeevat, H., E. Klein, and J. Calder. 1986. Unification Categorial Grammar. Centre for Cognitive Science, University of Edinburgh.