Squibs and Discussions

Parsing and Empty Nodes

Mark Johnson*
Brown University

Martin Kay†
Stanford University and Xerox PARC

This paper describes a method for ensuring the termination of parsers using grammars that freely posit empty nodes. The basic idea is that each empty node must be associated with a lexical item appearing in the input string, called its sponsor. A lexical item, as well as labeling the node for the corresponding word, provides labels for a fixed number, possibly zero, of empty nodes. The number of nodes appearing in the parse tree is thus bounded before parsing begins. Termination follows trivially. The technique is applicable to any standard parsing algorithm.

1. Introduction

One way of guaranteeing that a parsing algorithm will terminate is to ensure that each step consumes some finite amount of the input. There are two main situations in which this does not automatically occur, both arising from properties of the grammar. The first comes from nonbranching dominance chains of unbounded length. The second comes from empty nodes. Most modern grammars do not admit unbounded nonbranching chains, so that the problem of handling the phenomenon in parsing does not arise in practice. It is widely believed that these grammars also do not admit unbounded numbers of empty nodes. However, these generally constitute a problem in the design of parsing algorithms because the parser's domain of locality does not coincide with that of the constraints that govern their appearance.

This paper presents a proposal for constraining the appearance of empty nodes that is applicable to a wide variety of parsing strategies and linguistic theories, including many of those within the GB framework. Ideas like the ones to be presented here have been a part of other parsing systems, e.g., Fong (1991a, 1991b) and Millies (1991), and our notion of *sponsorship*, which we introduce below, can be viewed as a weak version of *lexicalization* in TAGs that is specifically focused on determining the distribution of empty nodes. The novelty of our approach lies principally in the identification of a single simple constraint as sufficient to ensure termination of the process. While its motivation is computational, its justification is primarily linguistic. The next section presents the problem that empty nodes pose for standard parsing techniques. Section 3 introduces the notion of *sponsorship*, and Section 4 discusses linguistic examples that demonstrate the role we see it playing. Section 5 shows how this proposal might be integrated into general parsing strategies. The conclusion summarizes what has been achieved, suggests avenues for further development, and draws parallels with some different approaches.

* Cognitive and Linguistic Sciences, Box 1978, Brown University, Providence, RI. E-mail: Mark_Johnson@brown.edu
† Xerox Parc, 333 Coyote Hill Rd., Palo Alto, CA 94304.

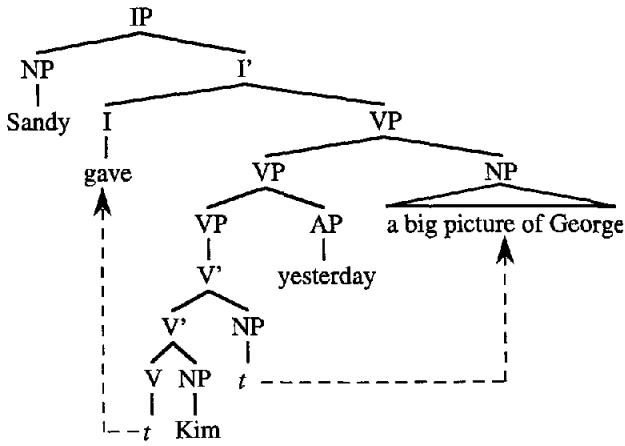

Figure 1
Extraposition and verb movement.

2. The Problem with Empty Nodes

The empty-node problem arises for the following reason. Given a parsing scheme with a standard notion of locality, there is no limit on the number of empty nodes that it might be necessary to posit before a configuration emerges in which the constraints governing the appearance of any of them can be verified.

We claim that most standard parsing algorithms will face difficulties in constraining the appearance of empty nodes in structures like the one in Figure 1.

A bottom-up parser would have to consider the possibility that every V′ should be combined with a following empty NP, like the upper V′ in Figure 1, to form another V′, which could then be treated in like manner. If the subcategorization frame of the V head were available, it could be used to bound the number of V′ nodes posited. But, in a structure involving verb movement, the head of the V chain is only integrated onto the structure after all of the V′ nodes have been constructed, so its subcategorization frame is not available when the V′ nodes are constructed. Similarly, the head of the NP chain is integrated too late to constrain the positing of V′ nodes.

A top-down parser would fare no better because the example is a classic case of left recursion. It might be argued that a top-down parser would have encountered the I head of the V chain before beginning to construct the V′ nodes and could therefore use its subcategorization frame to determine how many to construct. However, this would require an analysis of the grammar that is beyond the scope of standard parsing procedures. Notice that the V trace from which the subcategorization frame is projected is incorporated into the structure only after all the of V′ nodes have been constructed. Finally, the number of VP nodes is not determined by the subcategorization frame. No amount of grammar analysis will allow a top-down parser to predict the number of adjuncts attached to VP.

A head-first parser (Kay 1989; van Noord 1993) seems best adapted to the treatment of empty nodes. This mixed parsing strategy in effect predicts a head top-down and builds its complements and specifiers bottom-up. The trace of the verb would be identified immediately after the I *gave* had been recognized, since that trace is the

head of the complement of the I. But it is not clear how such a strategy would cope with empty nodes that do not stand in a head-to-head relationship, such as the trace associated with the adjoined NP. The construction of the NP *a big picture of George* would take place only after that of all of the V′ nodes to its left.

In summary, all of these parsing strategies suffer from the problem that they can posit too many—in some cases infinitely many—empty nodes. They do this because there is no limit on the number of empty nodes that can be posited before the constraints governing their appearance are verified.

Sometimes relatively simple strategies suffice to constrain the appearance of empty nodes and ensure parser termination. For example, given a grammatical constraint that all empty nodes be siblings of appropriate lexical heads, then simply delaying the introduction of an empty node until the node that dominates it is constructed suffices to constrain the number of empty nodes that a bottom-up parser posits. Similarly, for some theories of filler-gap dependencies, notably those based on 'slash features' (Gazdar, Klein, Pullum, and Sag 1985; Pereira 1981), it is possible to use a kind of prediction to constrain the possible occurrences of empty nodes in a wide variety of parsing strategies. However, with more complex theories of grammar, such as those within the GB framework, it is no longer so clear how, or even if, these sorts of techniques can be applied.

3. Sponsoring

Our solution to this problem is a device inspired by the notion of licensing in GB (Abney 1986). According to this conception, the presence and location of each empty node is justified by the specific structural relations it stands in with other nodes. For example, every noun phrase might be required to receive Case and a θ-role, and it may be that the phrase would have to appear at different places in the structure for both of these assignments to be made. However, it may also be that the phrase can be represented in one, or both, positions by a related empty category, a *trace* of the phrase, which is licensed by its fulfillment of this specific role.

To guarantee that only a finite number of empty nodes is posited in any analysis, we propose that, whatever parsing strategy is used, there be a global constraint on the number of empty nodes that can be posited in any single search path. We require that every empty node be *sponsored* by some lexical or morphological item that appears in the input. By sponsoring we mean that every empty node is associated with some nonempty lexical item, which we call its *sponsor*, and that the number of empty nodes that a single lexical token can sponsor is fixed by the lexicon, so that the set of all empty nodes to appear in the parse can be determined directly by a simple inspection of the lexical items in the input string.

Sponsorship is closely related to lexicalization in TAGs and CFGs (Schabes 1990, 1992; Schabes, Abeillé, and Joshi 1988; Schabes and Waters 1993; Vijay-Shanker and Schabes 1992). In a lexicalized grammar every node in the parse tree originates from some lexical entry, so parsing becomes a jigsaw puzzle–like problem of finding a consistent way of assembling the pieces of trees associated with each lexical item. Sponsoring is a weaker notion, in that only some of the constituent structure, namely the lexical items and empty nodes, are specified in lexical entries. This seems plausible in a framework in which general principles of grammar (e.g., X′ theory, Case theory, etc.) determine the overall structure of the parse tree. In addition, finding an appropriate association of constituent structure nodes with lexical items can be a difficult task. Because the sponsoring approach is only concerned with empty nodes, it should be easier to apply it to a wider variety of grammatical theories than a lexicalization

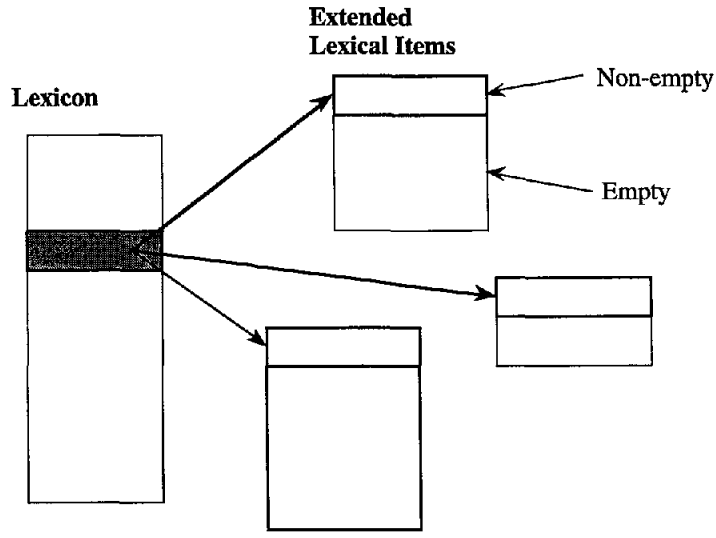

Figure 2
The structure of the lexicon and ELIs.

approach, which requires that every node (empty or not) be associated with a lexical item (but see the remarks in the conclusion below).

We now discuss one way to formulate a sponsoring approach. A lexical item and the set of empty categories that it sponsors constitute an *extended lexical item* (ELI) as sketched in Figure 2. In simple systems, such as the parser described in the next section, each lexical and morphological entry explicitly specifies the traces that it sponsors, but in more sophisticated implementations this could be determined automatically from principles of the grammar and properties of the lexical entry. It is not intended that sponsoring be used to change grammar, but only to impose relatively weak global constraints on the appearance of empty categories.

There are several variants of the basic idea. For example, one could require that *all* the empty nodes supplied by the lexicon be used in the analysis. On the one hand, this could lead to a proliferation of lexical entries. On the other, it could prune the search space more effectively if the role of each empty node were made as specific as possible.

As we remarked, previous proposals have made the number of empty nodes posited a function of the length of the input string. The novelty of our proposal is twofold. First, it provides a finer way of estimating the number of empty nodes that will occur. In fact, in the simplest version of the theory, the number of empty and nonempty terminals in a sentence is simply the sum of the sizes of the ELIs of the words in it. The number of empty categories is therefore this number minus the number of words. The fact that the number of empty nodes is bounded before parsing begins is the most important part of our proposal.

Our second proposal is that each of the items in an ELI is marked to show the specific role it must fill. Only one member, for example, will be capable of receiving a θ-role, and this member will not be capable of filling any position in which a θ-role is not assigned.

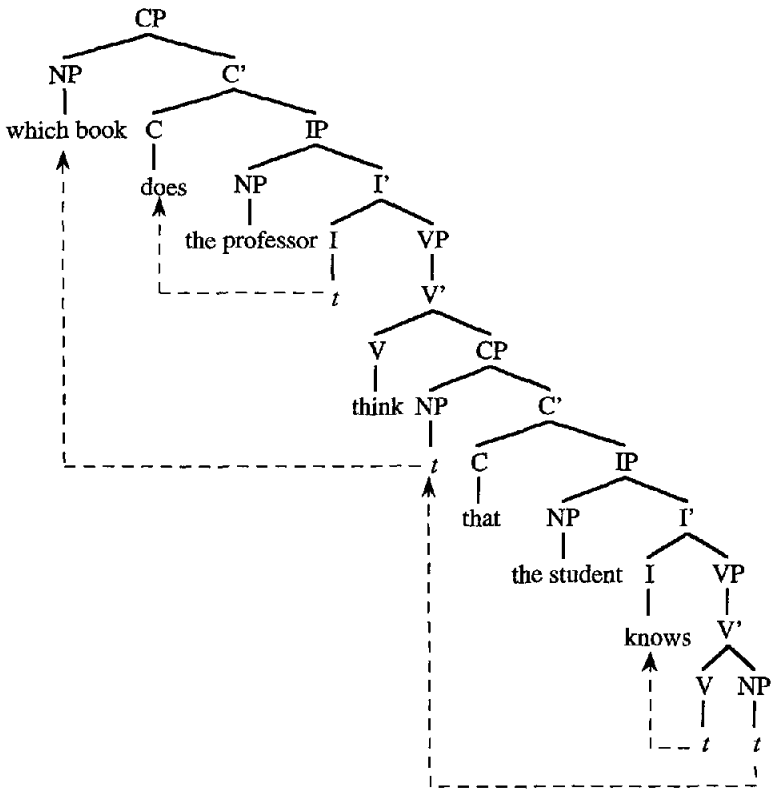

Figure 3
Cyclic WH-movement in English.

4. Linguistic Aspects of Sponsoring

The goal of this section is to demonstrate that the constraints that sponsoring imposes are plausible with respect to current linguistic assumptions. To the extent that they are, an important step will have been taken in establishing the decidability of these theories.

Consider once again the example in Figure 1. Because there is a government relationship between the V trace and the NP trace, and a movement relationship between the *gave* node under I and the V trace, it seems reasonable to include both of these traces in the ELI that permits *gave* to appear under I. The alternative clearly exists of allowing every N to sponsor an NP trace to allow, for example, for heavy NP shift of its maximal projection. It does not matter that this would lead to the provision of empty NP nodes in cases where no movement could occur, because the structures produced by the parser must independently satisfy all the requirements of the grammar.

Now consider an example involving cyclic WH-movement, as depicted in Figure 3. For English, WH-items such as *which* could sponsor the NP trace at the base of the chain (in Figure 3 the NP trace in embedded object position). However, we have already motivated a trace for the subcategorized complement, which should also serve as the foot of the WH-chain. Sponsors must also be found for the intermediate traces.

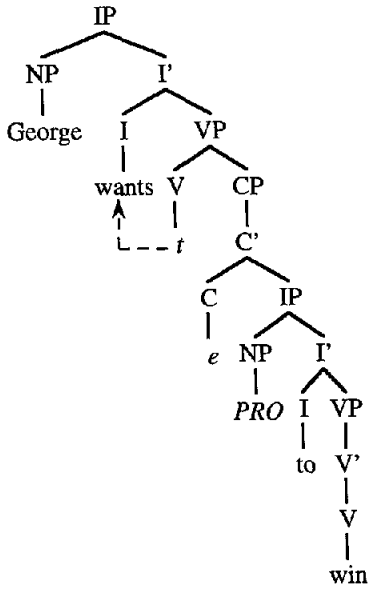

Figure 4
Empty C and PRO in English.

Because the number of intermediate traces that can appear in WH-constructions is not bounded, these intermediate traces cannot be sponsored by either the WH-item or the embedded verb. However, they can be sponsored by the bridge verbs that govern their CP parents. For example, the intermediate NP trace in Figure 3 is sponsored by the verb *think*.

Another possibility, inspired by the work of Grimshaw (1990) and Speas and Fukui (1986) on extended projections, is that inflection sponsors a complete set of empty functional nodes (and their specifiers) that can appear in the clause. In this example, the intermediate trace would be sponsored by the inflection *-s* on *knows*. While the first approach is perhaps more elegant, the second one also covers relative clauses, as discussed below. Either way, each potential location of an intermediate trace will have a sponsor; it is the independent constraints on WH-movement that are responsible for ensuring that, if a trace appears, it will be properly incorporated into a WH-chain.

The verb movement in Figure 3 can be treated as before. Presumably the ELI for *does* that permits it to appear under C also sponsors the corresponding trace in I, and the ELI for *knows* (or perhaps for the inflectional ending *-s*) that permits the verb to appear under I also sponsors the trace in V.

Next, consider the example in Figure 4. As well as the by now familiar V to I movement, it also exhibits two examples of empty categories that are not members of chains, so their sponsors cannot be determined on this basis. Responsibility for sponsorship of the empty C as well as the PRO could be ascribed to the verb *wants* that governs the CP in which they appear. This is a control verb and is in any case responsible for identifying *George* as the antecedent of the PRO. According to this view the inflected verb *wants* (i.e., the lexical stem and the inflection) sponsors a total of three empty categories. Alternatively, one could allow the infinitive marker *to* to sponsor PRO and the empty complementizer. This approach is consistent with the

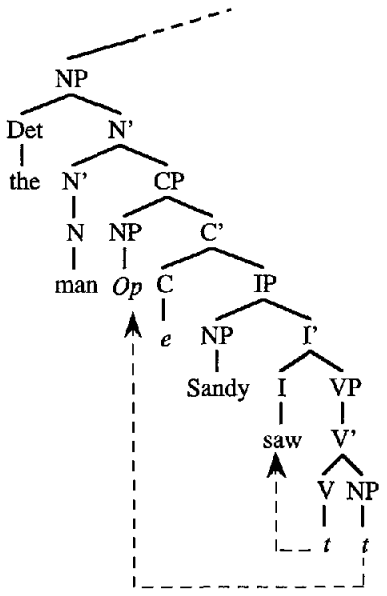

Figure 5
Empty operators in relative clauses.

view that inflection sponsors all of the empty functional nodes of the clause in which it appears.

English relative clauses are a major challenge for the sponsoring approach. Even though relative clauses share many of the structural properties of WH-question constructions such as cyclic movement, they can appear in a greater diversity of forms. All we attempt here is a survey of the problems encountered in developing the sponsoring account of empty nodes in relative clause constructions and their possible solutions.

Consider the case of a relative clause introduced by an empty operator *Op* (rather than an overt relative pronoun), such as the example in Figure 5.

The analyses discussed above provide sponsors for every empty node except the empty relative operator *Op* in the specifier position of CP. Because the number of relative clauses introduced by empty operators is not bounded (examples such as Example 1 seem to be indefinitely iterable) we are effectively forced to the conclusion that inflection, or some other lexical element present inside each relative clause, sponsors the empty operator *Op* in examples such as Example 1 and Figure 5.

Example 1
A man [Op_1 Kim likes t_1], [Op_2 Sandy hates t_2] ... and [Op_3 Ivan ignores t_3] ...

Even though the structure of reduced relative clauses such as Example 1 is not as well understood as ordinary relatives, they presumably involve empty operators as well. Assuming that we analyze the passive morphology on the participle as inflection (this seems linguistically motivated if we assume that movement to subject position is A-movement), the empty operator and all of its traces would be appropriately licensed.

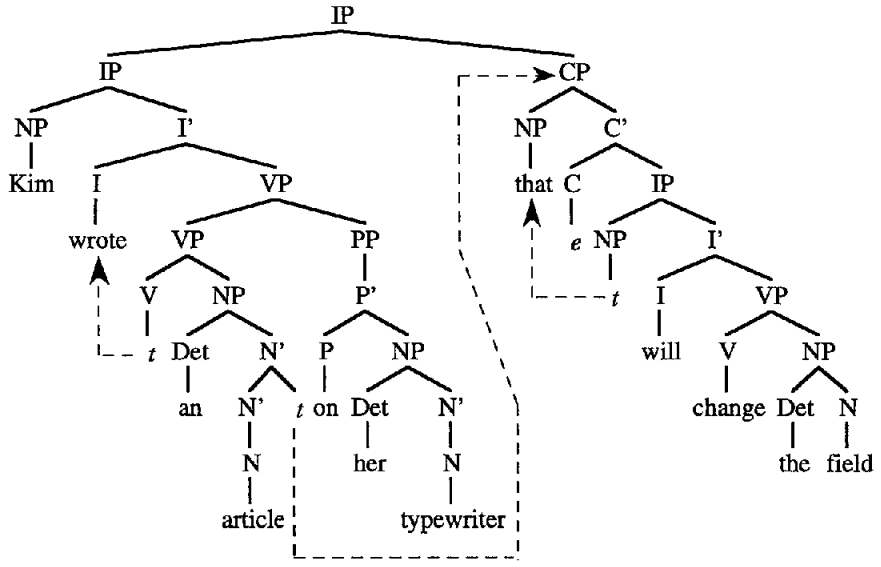

Figure 6
Relative clause extraposition.

Example 2
A horse [$_{CP}$*Op*[$_{IP}$*t*[$_{VP}$ ridden *t*] past the barn]] fell.

Finally, relative clauses can extrapose quite freely, as in Figure 6. (This diagram assumes that extraposed relative clauses adjoin to IP, but nothing rests on this assumption.)

The sponsoring mechanisms discussed above account for all of the empty nodes except for the trace of the extraposed relative clause (adjoined to N′ in Figure 6). As an anonymous *Computational Linguistics* reviewer points out, an apparently unbounded number of relative clauses can be extraposed from a single NP.

Example 3
A [$_{N'}$[$_{N'}$[$_{N'}$ photo t_1]t_2]t_3] appeared in today's paper [$_{CP_3}$ taken by Mapplethorpe] [$_{CP_2}$ showing him smiling] ... [$_{CP_1}$ that I think you would like].

Just as in the case of empty operator relative clauses, this iterability suggests that the extraposition traces must be sponsored by lexical items that appear inside the extraposed relative clauses. The inflection element in the relative clause seems to be the most appropriate lexical item to sponsor these traces.

To summarize, it seems that it is possible to identify sponsors for the empty nodes for a variety of linguistic constructions. Of course, the above examples do not demonstrate that it will always be possible to identify appropriate sponsors. In any given theory a detailed analysis of each construction is required to determine the appropriate sponsors.

5. Implementation

The diagram in Figure 7 shows the structure of a possible implementation of a parsing system that exploits the notion of sponsoring. Square cornered boxes are used for data, and round corners for processes. Lexical access is applied to the input string to produce (nondeterministically) the extended lexical item (ELI) of each word. Its output is split into a sequence of lexical items and a bag of empty nodes.

The parser can be based on any standard algorithm. It is special only in that all the terminal items in the phrases it constructs come either from the string of lexical items or from the bag of empty nodes, so it is impossible for any empty node to appear more than once in an analysis.

An obvious defect of the architecture in this simple form is that, in the absence of some form of prediction, the parser will consider at all points in the string all the structures that can be built entirely from empty nodes. A simple solution to this problem is to compute all the trees consisting solely of empty nodes sponsored by lexical items appearing in the utterance to be parsed before beginning the main parsing process. This will make it possible to use a parser that does not deal with empty nodes explicitly. The idea is to modify the interface between the parser and the grammar. The fact that sponsoring can be implemented entirely within the "rule-maker" interface shows that it can be used with any parsing algorithm. We take it that the main job of this interface will be to manufacture "rules" that enshrine the local well-formedness constraints on individual nodes. The modification consists in adding rules to this set.

A rule $a \rightarrow b_1 \ldots b_n$ will be passed in its original form to the parser, which can use it to build a phrase from n nonempty daughters. In addition, the rule maker supplies the parser with rules derived from this one by replacing $k < n$ of the b_i with empty trees, yielding a rule with n–k items on the right-hand side. The parser treats all these rules on equal footing. Apart from the sponsoring relationship, there is no requirement that any of the k empty trees be related to the n–k nonempty trees that the parser proper gets to see.

There are certain bookkeeping tasks that can best be undertaken by the rule maker. The most important of these have to do with ensuring that no empty terminal appears more than once in any structure. Concomitantly, it makes it possible to verify, at the end of the parse, that all empty terminals appear in the structure, if this is a requirement. The rule maker can also be used to percolate constraints up or down the tree, possibly discharging some of them in the process.

One remaining problem is to arrange for the parser to include the empty trees in the structures that it builds. We assume that this information accompanies the

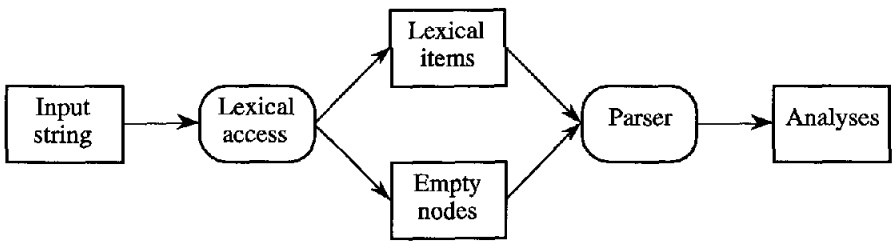

Figure 7
A simple architecture.

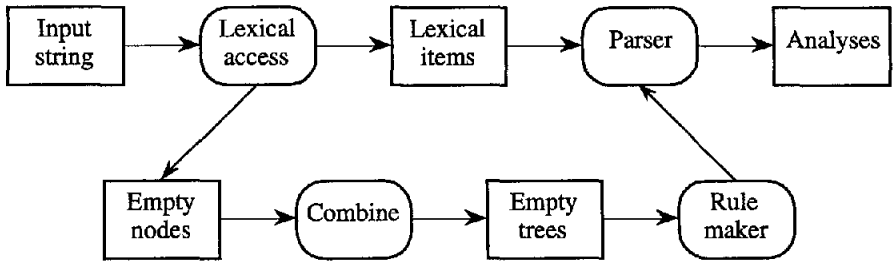

Figure 8
The modified architecture.

rule, perhaps in the form of features on the parent category, as in many unification grammars.

6. Conclusions

It has not been our purpose here to solve the problem of parsing for GB, but only to provide a mechanism for ensuring that empty nodes do not cause nontermination of parsing in an important class of cases. We have made only very general remarks on the architecture of a parsing system that would incorporate these ideas, largely because we believe that the details of such a design would depend heavily on the mechanism that was chosen for managing constraints. Efficient implementation would depend on a good resolution of a number of interacting trade-offs, and there are several of these within our scheme that need to be explored. In particular, the components of an ELI could be more or less narrowly specified for the roles they are to fill. If the nodes are highly specialized, there will be greater ambiguity in the lexicon and consequently greater nondeterminism in the parser. On the other hand, many of these search paths will presumably be curtailed earlier than they would have been with less specialized nodes.

A major determinant of system performance will clearly be the manner in which constraints are enforced. It is possible to distinguish a class of constraints that arise in the course of parsing but which cannot, in general, be discharged there, and should therefore be treated as part of the result that the parser delivers. Notable among these are contraindexing constraints from the Binding theory.

Ensuring that each node in an ELI fills the role for which it was intended could be resolved through the general constraint mechanism. However, more specialized mechanisms could sometimes be useful. Suppose, for example, that the lexical entry for a noun contained a node specifically intended to receive Case. If these were the only nodes whose Case attribute was unspecified, all others having an explicit zero value, the required mechanism could consist simply in having all rules assign a value to this feature, that value being zero except for rules that assign a substantive Case.

A somewhat different problem consists of verifying that nodes from a given ELI appear in a certain structural configuration. Assigning each node a unique identifier allows this problem to be solved straightforwardly by the general constraint mechanism.

It might be advantageous for the ELI to encode very specific information about a lexical item and the empty nodes that it sponsors. For example, the ELI for a WH

item might specify that the traces it sponsors are coindexed with the WH item itself. Assuming that indices are just unbound variables (thus coindexing is unification and contraindexing is an inequality constraint), an interesting technical problem arises if the basic parsing engine uses a chart (Kay 1967, 1980). Because it is fundamental to such devices that the label on an edge is copied before it is used as a component of a larger phrase, the variables representing indices will be copied or renamed and the indices on the WH item and its sponsored trace will no longer be identical. However, it is important that the sharing of variables among the components of an ELI be respected when they come together in a phrase. One way of overcoming this problem is to associate a vector of variables with each edge, in which each variable that is shared between two or more edges is assigned a unique position. Whenever edges are combined their associated vectors are unified, thus ensuring that the corresponding variables in each edge are identified.

Finally, our linguistic examples suggest to us that a more focused notion of sponsoring might be formulated. We observe that, modulo adjunction, empty nodes tend to stand in fixed structural relations to their sponsors. If this is indeed generally true, then these strong locality constraints should clearly be exploited in the parsing process. This amounts to adopting the framework of Tree Adjoining Grammars (Frank 1990; Joshi, Levy, and Takahashi 1975; Kroch and Joshi 1985; Schabes 1990). The emphasis would then fall on deriving the initial and auxiliary trees from the general principles of grammar.

Acknowledgments

This research was supported by the Institut für maschinelle Sprachverarbeitung at the University of Stuttgart. We would like to thank Professor Christian Rohrer and the members of the Institut for providing us with this opportunity. We are also indebted to Lauri Karttunen and two anonymous *Computational Linguistics* reviewers for their helpful comments during the preparation of this paper.

References

Abney, Steven (1986). "Licensing and parsing." *North Eastern Linguistic Society* 17, 1–15.

Fong, Sandiway (1991a). "The computational implementation of principle-based parsing." In *Principle-Based Parsing: Computation and Psycholinguistics*, edited by Robert C. Berwick, Steven P. Abney, and Carol Tenny, 39–65. Kluwer Academic Publishers.

Fong, Sandiway (1991b). *Computational properties of principle-based grammatical theories*. Doctoral dissertation, Massachusetts Institute of Technology.

Frank, Robert (1990). *Computation and linguistics theory: A government binding parser using tree adjoining grammar*. Master's dissertation, University of Pennyslvania.

Gazdar, Gerald; Klein, Ewan; Pullum, Geoffrey; and Sag, Ivan (1985). *Generalized Phrase Structure Grammar*. Blackwell.

Grimshaw, Jane (1990). *Argument Structure*. MIT Press.

Joshi, Aravind K.; Levy, L. S.; and Takahashi, M. (1975). "Tree adjunct grammars." *Journal of Computer and System Sciences*, 10(1).

Kay, Martin (1967). "Experiments with a powerful parser." *2ème Conference International sur le traitment automatique des languages*, Grenoble, France.

Kay, Martin (1980). "Algorithm schemata and data structures in syntactic processing." In *Readings in Natural Language Processing*, edited by Barbara J. Grosz, Karen Sparck Jones, and Bonnie Lynn Weber, 35–70. Morgan Kaufmann.

Kay, Martin (1989). "Head-driven parsing." In *Proceedings, 1st International Workshop on Parsing Technologies*, Pittsburgh, PA, 52–62.

Kroch, Anthony, and Joshi, Aravind K. (1985). "Linguistic relevance of tree adjoining grammars." Technical Report MS-CIS-85-18, Department of Computer and Information Science, University of Pennsylvania.

Millies, Sebastian (1991). "Modularity, parallelism and licensing in a principle-based parser for German." CLAUS Report Nr. 17, Computerlinguistik, Universität des Saarlandes.

Pereira, Fernando C. N. (1981). "Extraposition grammars." *Computational Linguistics*, 7(4), 243–256.

Schabes, Yves (1990). *Mathematical and computational aspects of lexicalized grammars*. Doctoral dissertation, University of Pennsylvania.

Schabes, Yves (1992). "Stochastic lexicalized tree-adjoining grammars." In *Proceedings, Fifteenth International Conference on Computational Linguistics (COLING–92)*. Nantes, France, 426–432.

Schabes, Yves; Abeillé, Anne; and Joshi, Aravind K. (1988). "Parsing strategies with 'lexicalized' grammars: application to tree adjoining grammars." In *Proceedings, 12th International Conference on Computational Linguistics*. Budapest, Hungary.

Schabes, Yves, and Waters, Richard C. (1993). "Lexicalized context-free grammars." In *Proceedings, 31st Annual Meeting of the Association for Computational Linguistics*. Columbus, Ohio, 121–129.

Speas, M., and Fukui, N. (1986). "Specifiers and projections." In *MIT Working Papers in Linguistics 8*, Department of Linguistics and Philosophy, MIT, Cambridge, MA.

van Noord, Gertjan (1993). *Reversibility in natural language processing*. Doctoral dissertation, University of Utrecht.

Vijay-Shanker, K., and Schabes, Yves (1992). "Structure sharing in lexicalized tree-adjoining grammars." In *Proceedings, 15th International Conference on Computational Linguistics (COLING–92)*. Nantes, France, 205–211.