Phrase Structure Trees Bear More Fruit than You Would Have Thought[1]

Aravind K. Joshi

**Department of Computer and Information Science
The Moore School/D2
University of Pennsylvania
Philadelphia, PA 19104**

Leon S. Levy

**Bell Telephone Laboratories
Whippany, NJ 07981**

In this paper we will present several results concerning phrase structure trees. These results show that phrase structure trees, when viewed in certain ways, have much more descriptive power than one would have thought. We have given a brief account of local constraints on structural descriptions and an intuitive proof of a theorem about local constraints. We have compared the local constraints approach to some aspects of Gazdar's framework and that of Peters and Ritchie and of Karttunen. We have also presented some results on skeletons (phrase structure trees without labels) which show that phrase structure trees, even when deprived of the labels, retain in a certain sense all the structural information. This result has implications for grammatical inference procedures.

1. Introduction

There is renewed interest in examining the descriptive as well as generative power of phrase structure grammars. The primary motivation for this interest has come from the recent investigations in alternatives to transformational grammars (e.g., Bresnan 1978; Kaplan and Bresnan 1979; Gazdar 1978, 1979a, 1979b; Peters 1980; Karttunen 1980).[2] Some of these approaches require amendments to phrase structure grammars (especially Gazdar 1978, 1979a, 1979b; Peters 1980; Karttunen 1980) that increase their descriptive power without increasing their generative power. Gazdar wants to restrict the power to that of context-free languages. Others are not completely precise on this aspect. Berwick has shown that the Kaplan-Bresnan system is nearly equivalent to the

[1] This work was partially supported by NSF grant MCS 79-08401 and MCS 81-07290.

This paper is a revised and expanded version of a paper presented at the 18th Annual Meeting of the Association for Computation Linguistics, University of Pennsylvania, Philadelphia, June 1980.

Thanks are extended to the two referees of this paper and to Michael McCord for their valuable comments, which helped in the improvement of both the content and the presentation of the material herein.

[2] Since this paper was submitted for publication, a number of papers have appeared that should be of interest to its readers. "Phrase Linking Grammars" by S. Peters and R.W. Ritchie describes their system (Technical Report, Department of Linguistics, University of Texas at Austin, 1982). Strong adequacy of context-free grammars has been discussed by J. Bresnan, R.M. Kaplan, S. Peters, and A. Zaenan in "Cross-Serial Dependencies in Dutch" (to appear in *Linguistic Inquiry* in 1982). This paper shows that context-free grammars are not strongly adequate (i.e., they are unable to provide the appropriate structural descriptions) to characterize cross-serial dependencies in Dutch. In a recent paper ("How much context-sensitivity is needed to provide reasonable structural descriptions: A tree adjoining system for generating phrase structure trees," presented at the Parsing Workshop, Ohio State University, May 1982; also Technical Report, Department of Computer and Information Science, University of Pennsylvania, 1982), A. Joshi discusses weak and strong adequacy of grammars and proposes a tree adjoining grammar (TAG) that appears to be strongly adequate and has only slightly more power than context-free grammars. Joshi has also given a rough characterization of a class of context-sensitive language (MCSL) that appears to be suitable to characterize natural languages. Languages of TAG belong to MCSL, and languages of PLG's of Peters and Ritchie also belong to this class. (TAG's use a linking device similar to that in the PLG's.)

0362-613X/82/010001-11$01.00

so-called indexed grammars. The power of the phrase linking grammars of Peters and Ritchie is not completely known at this time.

The notion of node admissibility plays an important role in these formulations. The earliest reference to node admissibility appears in Chomsky 1965 (p. 215); he suggests the possibility of constructing a rewriting system where the rewriting of a symbol is determined not only by the symbol being rewritten but also by the dominating category symbol. In his analysis of the base component of a transformation grammar, McCawley 1968 suggested that the appropriate role of context-sensitive rules in the base component of a transformational grammar can be viewed as node admissibility conditions on the base trees. The base component is thus a set of labeled trees satisfying certain conditions. Peters and Ritchie 1969 made this notion precise and proved an important result, which roughly states that the weak generative power of a context-sensitive grammar is that of a context-free grammar, if the rules are used as node admissibility conditions. Later Joshi and Levy 1977 made a substantial extension of this result and showed that, if the node admissibility conditions include Boolean combinations of proper analysis predicates and domination predicates, the weak generative capacity is still that of a context-free grammar.

Besides the notion of node admissibility, Gazdar introduces two other notions in his framework (Generalized Phrase Structure Grammars, GPSG). These are (1) categories with holes and an associated set of derived rules and linking rules, and (2) metarules for deriving rules from one another. The categories with holes and the associated rules do not increase the weak generative power beyond that of context-free grammars. The metarules, unless constrained in some fashion, will increase the generative power, because, for example, a metarule can generate an infinite set of context-free rules that can generate a strictly context-sensitive language. (The language $\{a_n b_n c_n / n \geq 1\}$ can be generated in this way.) The metarules in the actual grammars written in the GPSG framework so far are constrained enough so that they do not increase the generative power.

Besides node admissibility conditions, Peters 1980 introduces a device for "linking" nodes (see also Karttunen 1980). A lower node can be "linked" to a node higher in the tree and becomes "visible" while the semantic interpretation is carried out at the lower node. The idea here is to let the context-free grammar overgenerate and the semantic interpretation weed out ill-formed structures. Karttunen 1980 has developed a parser using this idea.

Kaplan and Bresnan 1979 have proposed an intermediate level of representation called functional structure. This level serves to filter structures generated by a phrase structure grammar. Categories with holes are not used in their framework. In this paper we will not be concerned with the Kaplan-Bresnan system.

In Section 2 we briefly review Gazdar's proposal, especially his notion of categories with holes. We give a short historical review of this notion.

In Section 3 we briefly describe our work on local constraints on structural descriptions (Joshi and Levy 1977; Joshi, Levy, and Yueh 1980). We give an intuitive explanation of these results.

In Section 4 we propose some extensions of our results and discuss them in the context of some long distance rules. We also describe Peters's 1980 approach and present some suggestions for "context-sensitive" compositional semantics.

In Section 5 we briefly present the framework of Peters and Karttunen and compare it with that of Gazdar and of ourselves.

In Section 6 we briefly discuss our results concerning a characterization of structural descriptions entirely in terms of trees without labels.

2. Gazdar's Formulation

Gazdar 1979 has introduced categories with holes and some associated rules in order to allow for the base generation of "unbounded" dependencies. Let V_N be the set of *basic* nonterminal symbols. Then we define a set $D(V_N)$ of *derived* nonterminal symbols as follows.

$$D(V_N) = \{\alpha/\beta \mid \alpha, \beta \in V_N\}$$

For example, if S and NP are the only two nonterminal symbols, then $D(V_N)$ would consist of S/S, S/NP, NP/NP, and NP/S. The intended interpretation of a derived category (slashed category or a category with a hole) is as follows: A node labeled α/β will dominate subtrees identical to those that can be dominated by α, except that somewhere in every subtree of the α/β type there will occur a node of the form β/β dominating a resumptive pronoun, a trace, or the empty string, and every node linking α/β and β/β will be of the form σ/β. Thus α/β labels a node of type α that dominates material containing a hole of the type β (i.e., an extraction site in a movement analysis). For example, S/NP is a sentence that has an NP missing somewhere. The derived rules allow the propagation of a hole and the linking rules allow the introduction of a category with a hole. For example, given the rule (1)

$$[_{S}\ \mathrm{NP\ VP}]\ ^{3} \qquad (1)$$

[3] This is the same as the rule
S → NP VP
but written as a node admissibility condition.

we will get two derived rules (2) and (3)

$$[_{S/NP}\ NP/NP\ VP] \qquad (2)$$

$$[_{S/NP}\ NP\ VP/NP] \qquad (3)$$

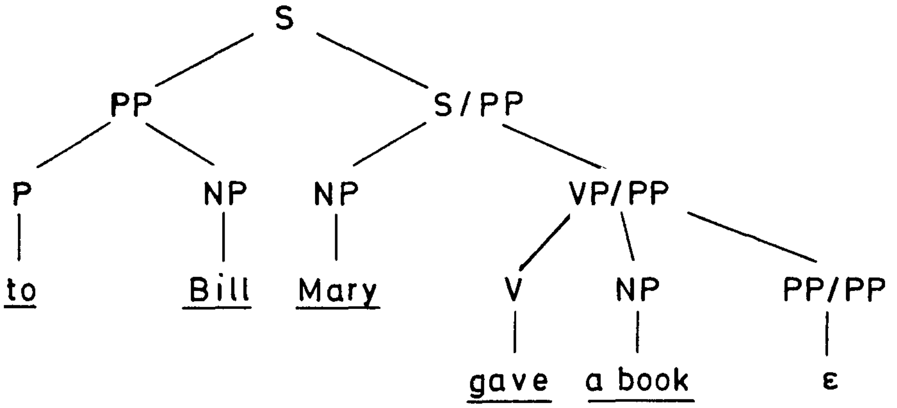
An example of a linking rule is a rule (rule schema) that introduces a category with a hole as needed for topicalization.

$$[_{S}\ \alpha\ S/\alpha] \qquad (4)$$

For α = PP this becomes

$$[_{S}\ PP\ S/PP\ ] \qquad (5)$$

This rule will induce a structure like (6). The technique of categories with holes and the associated derived and linking rules allows unbounded dependencies to be accounted for in the phrase structure representation; however, this is accomplished at the expense of proliferation of categories of the type α/β (see also Karttunen 1980). Later, in Section 3, we will present an alternate way of representing (6) by means of local constraints and some of their generalizations.



(6)

The notion of categories with holes is not completely new. In his 'String Analysis of Language Structure', Harris 1956, 1962 introduces categories such as S – NP or S_{-NP} (like S/NP of Gazdar) to account for moved constituents. He does not however seem to provide, at least not explicitly, machinery for carrying the "hole" downwards. He also has rules in his framework for introducing categories with holes. Thus, in his framework, something like (6) would be accomplished by allowing for a sentence form (a center string) of the form (7) (not entirely his notation).

$$NP\ V\ \Omega_{-NP} \qquad (7)$$

Ω = Object or Complement of V

Sager, who has constructed a very substantial parser starting from some of these ideas and extending them significantly, has allowed for the propagation of the 'hole' resulting in structures very similar to those of Gazdar. She has also used the notion of categories with holes in order to carry out some coordinate structure computation. For example, Sager allows for the coordination of S/α and S/α but not S and S/α. (See Sager 1967 for an early reference to her work.)

Gazdar is the first, however, to incorporate the notion of categories with holes and the associated rules in a formal framework for his syntactical theory and also to exploit it in a systematic manner for explaining coordinate structure phenomena.

3. Local Constraints

In this section we briefly review our work on local constraints. Although this work has already appeared (Joshi and Levy 1977, Joshi, Levy, and Yueh 1980) and attracted some attention recently, the demonstration of our results has remained somewhat inaccessible to many due to the technicalities of the tree automata theory. In this paper we present an intuitive account of these results in terms of interacting finite state machines.

The method of local constraints is an attempt to describe context-free languages in an apparently context-sensitive form that helps to retain the intuitive insights about the grammatical structure. This form of description, while apparently context-sensitive, is in fact context-free.

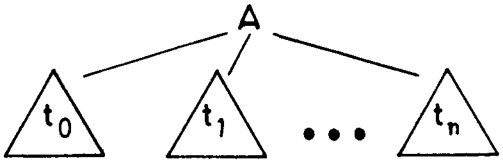
3.1 Definition of Local Constraints

Context-sensitive grammars, in general, are more powerful (with respect to weak generative capacity) than context-free grammars. A fascinating result of Peters and Ritchie 1969 is that if a context-sensitive grammar G is used for "analysis" then the language "analyzed" by G is context-free. First, we describe what we mean by the use of a context-sensitive grammar G for "analysis". Given a tree t, we define the set of *proper analyses* of t. Roughly speaking, a proper analysis of a tree is a slice across the tree. More precisely, the following recursive definition applies:

Definition 3.1. The set of proper analyses of a tree t, denoted Pt, is defined as follows.

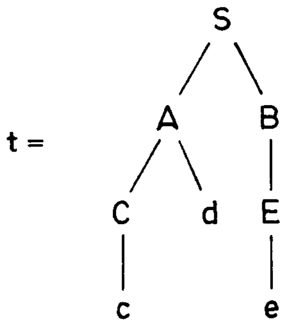
(i) If t = ϕ (the empty tree), then
Pt = ϕ.

(ii) If t =



then $Pt = \{A\} \cup P(t_0).P(t_1). \ldots .P(t_n)$
where t_0, t_1, ... t_n are trees, and '.' denotes concatenation (of sets).

Example 3.1



Pt = {S, AB, AE, Ae, CdB, CdE, Cde, cdB, cdE, cde}.

Let G be a context-sensitive grammar; i.e., its rules are of the form
$A \rightarrow \omega/\pi\_\_\phi$
where $A \in V - \Sigma$ (V is the alphabet and Σ is the set of terminal symbols), $\omega \in V^+$ (set of non-null strings on V) and $\pi, \phi \in V^*$ (set of all strings on V). If π and ϕ are both null, then the rule is a context-free rule. A tree t is said to be "analyzable" with respect to G if for each node of t some rule of G "holds". It is obvious how to check whether a context-free rule holds of a node or not. A context-sensitive rule $A \rightarrow \omega/\pi\_\_\phi$ holds of a node labeled A if the string corresponding to the immediate descendants of that node is ω and there is a proper analysis of t of the form $\rho_1\pi A\phi\rho_2$ that "passes through" the node, ($\rho_1,\rho_2 \in V^*$). We call the contextual condition $\pi\_\_\phi$ a *proper analysis predicate*.

Similar to these context-sensitive rules, which allow us to specify context on the "right" and "left", we often need rules to specify context on the "top" or "bottom". Given a node labeled A in a tree t, we say that $DOM(\pi\_\_\phi)$, $\pi, \phi \in V^*$, holds of a node labeled A if there is a path from the root of the tree to the frontier, which passes through the node labeled A, and is of the form
$\rho_1\pi A\phi\rho_2$ $(\rho_1,\rho_2 \in V^*)$.
The contextual condition associated with such a "vertical" proper analysis is called a *domination predicate*.

The general form of a local constraint combines the proper analysis and domination predicates as follows:

Definition 3.2. A local constraint rule is a rule of the form
$A \rightarrow \omega/C_A$
where C_A is a Boolean combination of proper analysis and domination predicates.

In transformational linguistics the context-sensitive and domination predicates are used to describe conditions on transformations; hence we have referred to these local constraints elsewhere as local transformations.

3.2 Results on Local Constraints

Theorem 3.1 (Joshi and Levy 1977) Let G be a finite set of local constraint rules and $\tau(G)$ the set of trees analyzable by G. (It is assumed here that the trees in $\tau(G)$ are sentential trees; i.e., the root node of a tree in $\tau(G)$ is labeled by the start symbol, S, and the terminal nodes are labeled by terminal symbols.) Then the string language $L(\tau(G)) = \{x \mid x \text{ is the yield of t and } t \in \tau(G)\}$ is context-free.

Example 3.2 Let V = {S,T,a,b,c,e} and Σ = {a,b,c,e}, and G be a finite set of local constraint rules:

1. $S \rightarrow e$
2. $S \rightarrow aT$
3. $T \rightarrow aS$
4. $S \rightarrow bTc \;/\; (a\_\_) \wedge \mathrm{DOM}\;(T\_\_)$
5. $T \rightarrow bSc \;/\; (a\_\_) \wedge \mathrm{DOM}\;(S\_\_)$

In rules 1, 2, and 3, the context is null, and these rules are context-free. In rule 4 (and in rule 5), the constraint requires an 'a' on the left, and the node dominated (immediately) by a T (and by an S in rule 5).

The language generated by G can be derived by G_1:

$$\begin{array}{ll} S \rightarrow e & S \rightarrow aT_1 \\ S \rightarrow aT & T \rightarrow aS_1 \\ T \rightarrow aS & T_1 \rightarrow bSc \\ S_1 \rightarrow bTc & \end{array}$$

In G_1 there are additional nonterminals S_1 and T_1 that enable the context checking of the local constraints grammar, G, in the generation process.

It is easy to see that, under the homomorphism that removes subscripts on the nonterminals T_1 and S_1, each tree generable in G_1 is analyzable in G. Also, each tree analyzable in G has a homomorphic preimage in G_1.

The methods used in the proof of the theorem use tree automata to check the local constraint predicates, since tree automata used as recognizers accept only tree sets whose yield languages are context-free.

We now give an informal introduction to the ideas of (bottom-up) tree automata. Tree automata process labeled trees, where there is a left-to-right ordering on the successors of a node in the tree. When all the successors of a node ν have been assigned states, then a state is assigned to ν by a rule that depends on the label of ν and the states of the successors of ν considering their left-to-right ordering. Note that the automaton may immediately assign states to the nodes on the frontier of the tree since these nodes have no successors. If the set of states is partitioned into final and non-final states, then a tree is accepted by the automaton if the state assigned to the root is a final state. A set of trees accepted by a tree automaton is called a *recognizable set.* Note that the automaton may operate non-deterministically, in which case, as usual, a tree is accepted if there is some set of state assignments leading to its acceptance.

The importance of tree automata is that they are related to the sets of derivation trees of context-free grammars. Specifically, if T is the set of derivation trees of a context-free grammar, G, then there is a tree automaton that recognizes T. Conversely, if T is the set of trees recognized by a tree automaton, A, then T may be systematically relabeled as the set of derivation trees of a context-free grammar.

The basic idea presented in detail in Joshi and Levy 1977 is that, because tree automata have nice closure properties (closure under union, intersection, and concatenation), they can do the computations required to check the local constraints.

Another way of looking at the checking of a labeled tree by a tree automaton is as follows. We imagine a finite state machine sitting at each node of a tree. The role of the finite state machine is to check that a correct rule application is made at the node it is checking. Initially, the nodes on the frontier are turned on and signal their parent nodes. At any other node in the tree, the machine at that node is turned on as soon as all its direct descendants are active. Assuming that at each node the machine for that node has checked that the rule applied there was one of the rules of the context-free grammar we are looking for, then when the root node of the tree signals that it has correctly checked the root we know that the tree is a proper tree for the given context-free grammar.

When checking for local constraints, a machine at a given node not only passes information to its parent, as described above, but also passes information about those parts of the local constraints, corresponding to the given node as well as all its descendants, that have not yet been satisfied. The point is that this information is always bounded and hence a finite number of states are adequate to code this information.

The fact that the closure properties hold can be seen as follows. Consider a slightly more general situation. We consider an A machine and a B machine at each node. Depending on the connections between these A and B machines, we obtain additional results. For example, as each A machine passes information to its parent, it may also pass information to the B machine, but the B machine will not pass information back to the A machine. The tree is accepted if the B machine at the root node of the tree ends up in a final state. Although this seems to be a more complicated model, it can in fact be subsumed in our first model and is the basis of an informal proof that the recognition rules are closed under intersection, since the A machine and the B machine can check different rules.

An important point is that the local constraint on a rule applied at a given node may only be verified by the checking automata at some distant ancestor of that node. In particular, in the case of a proper analysis constraint, it can only be verified at a node sufficiently high in the tree to dominate the entire string specified in the constraint.

The perceptive reader may now be wondering what replaces all these hypothetical finite state machines when the set of trees corresponds to a context-free grammar. Well, if we were to convert our local constraints grammar into a standard form context-free grammar, we would require a larger nonterminal set. In effect this larger nonterminal set is an encoding of the finite state information stored at the nodes.

The intuitive explanation presented in this section is, in fact, a complete characterization of recognizability. Given a context-free grammar, one can specify the finite state machine to be posted at each node of a tree to check the tree. And conversely, given the finite state machine description, one can derive the equivalent context-free grammar.

The essence of the local constraints formulation is to paraphrase the finite state checking at the nodes of the tree in terms of patterns or predicates.

4. Some Generalizations

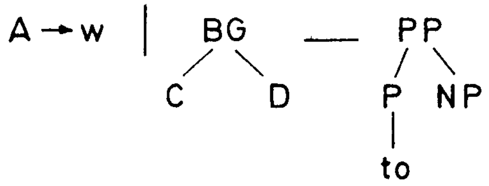
The result of Theorem 3.1 can be generalized in various ways. Generalizations in (i) and (ii) below are immediate.

(i) *Variables* can be included in the constraint. Thus, for example, a local constraint rule can be of the form

A → w | BCDXE __ FYG

where A,B,C,D,E,F,G are nonterminals, w is a string of terminals and/or nonterminals, and X and Y are variables that range over arbitrary strings of terminals and nonterminals.

(ii) *Finite tree fragments* can be included in the constraint. Thus, for example, a local constraint rule can be of the form



Another useful generalization has the following essential character.

(iii) *Predicates that relate nodes mentioned in the proper analysis predicates and domination predicates* (associated with a rule), as well as nodes in finite tree fragments dominated by these nodes, can be included in the constraint. Unfortunately, at this time we are unable to give a precise characterization of this generalization. The following two predicates are special cases of this generalization, and Theorem 3.1 holds for these two cases.

(a) *COMMAND*
COM (A B C)
B immediately dominates A and B dominates C, not necessarily immediately. Usually B is an S node.

(b) *LEFTMOSTSISTER*
LMS(A B)
A is the leftmost sister of B

Example 4.1

Let us consider (6) in Section 2 (topicalization).

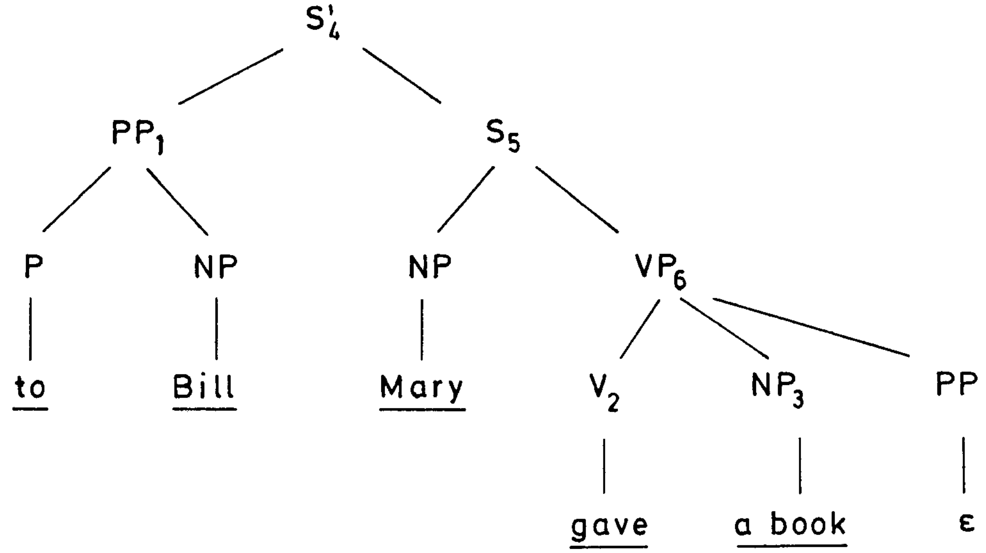
Consider the following fragment of a local constraints grammar, G. (Only some of the relevant rules are shown.)

S → NP VP
VP → V NP PP
S′ → PP S
.
.
.
V → give | __ NP PP
.
.
.
$PP \rightarrow \epsilon \mid PP_1\ X\ V_2\ NP_3$ __
$\wedge$ DOM $(S'_4\ S_5\ Y\ VP_6$ __ $)$
$\wedge$ COM $(PP_1\ S'_4\ VP_6)$
$\wedge$ LMS $(PP_1\ S_5)$
.
.
.

The last rule has a proper analysis predicate, a domination predicate, and COMMAND and LEFTMOSTSISTER predicates whose arguments satisfy the requirements mentioned in (iii) above (i.e., they relate nodes mentioned in proper analysis predicates and domination predicates). The indexing makes this clear.

Structure (7) will be well-formed. Compare (7) with (6) in Section 1 (Gazdar's framework) and with (8) in Section 5 (Peters's and Karttunen's framework).

(7)



4.1 Local Constraints in Semantics

Since a local constraint rule has a context-free part and contextual constraint part, it is possible to define context-sensitive compositional semantics in the following manner.

For a context-free rule of the form

A → BC

if σ(A), σ(B), σ(C) are the 'semantic' translations associated with A, B, and C, respectively, then σ(A) is a composition of σ(B) and σ(C).

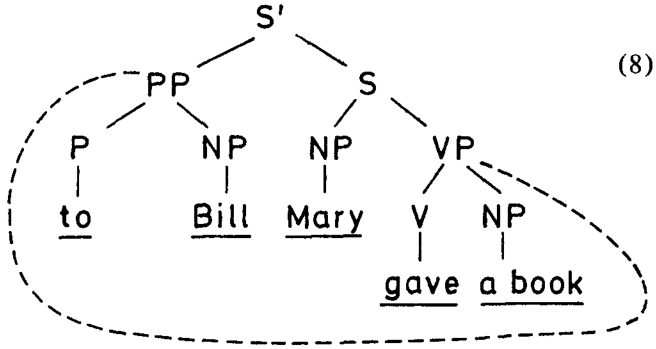
For a local constraint rule of the form

A → BC | P

where A → BC is the context-free part and P is the contextual constraint, we can have σ(A) as a composition of σ(B) and σ(C), which depends on P. This idea has been pursued in the context of programming languages (Joshi, Levy, and Yueh 1980). Whether such an approach would be useful for natural language is an open question. (An additional comment appears in Section 5.)

5. Linked Nodes[4] (Peters's and Kartunnen's framework)

Peters 1980 and Kartunnen 1980 have proposed a device for linking nodes to handle unbounded dependencies. Thus, for example, instead of (6) or (7), we have (8).



(8)

The dotted line that loops from the VP node back to the moved constituent is a way of indicating the location of the gap in the object position under the VP. The link also indicates that there is certain dependency between the gap and the dislocated element. Both in our approach and that of Peters and Karttunen, proliferation of categories as in Gazdar's approach is avoided. Further, for Peters and Karttunen, while carrying out bottom-up semantic translation, the moved constituent is "visible" at the VP node. In our approach, this "visibility" can be obtained if the translation is made to depend on the contextual constraint which, of course, has already been checked prior to the translation. This is the essence of our suggestion in Section 4.1.

Karttunen 1980 has constructed a parser incorporating the device of linked nodes. Karttunen also discusses the problem of complex patterns of moved constituents and their associated gaps or resumptive pronouns. This is not easy to handle in Gazdar's framework without multiplying the categories even further, e.g., by providing categories such as S/NP NP, etc.[5] Karttunen handles this problem by essentially incorporating the checking of the patterns of gaps and fillers in the parser, i.e., in the control structure of the parser.

Our approach can be regarded as somewhat intermediate between Gazdar's and that of Peters and Karttunen in the following sense. We avoid multiplication of categories as do Peters and Karttunen. On the other hand, the relationship between the moved constituent and the gap is expressed in the grammar itself (more in the spirit of Gazdar) instead of in the parser (more precisely, in the data structure created by the parser) as in the Peters and Karttunen approach.

We have not pursued the topic of multiple gaps and fillers in our framework but, obviously, in it we would opt for Karttunen's suggestion of checking the constraints on the patterns of gaps and fillers in the parser itself. It could not be done by local constraints alone because local constraints essentially do the work of the links in the Peters and Karttunen framework.

6. Skeletal Structural Descriptions (Skeletons)

In Section 4, we showed how local constraints allowed us to prevent proliferation of categories. We can dispense with the local constraints and construct an equivalent context-free grammar that would have potentially a very large number of categories. While pursuing the relation between 'structure' and the size of the nonterminal vocabulary (i.e., the syntactic categories), we were led to the following surprising result: the actual labels, in a sense, carry no information. (This result was also used by us in developing some heuristics for converting a context-free grammar into a more compact but equivalent local constraints grammar. We will not describe this use of our result in the present paper. (For further information, see Joshi, Levy, and Yueh 1980.)

First we need some definitions. A phrase structure tree without labels will be called a *skeletal structural*

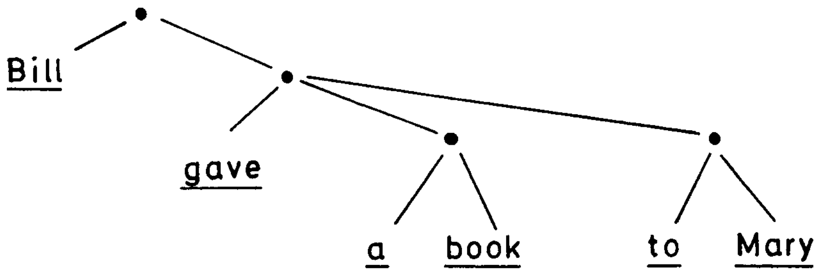
[4] We give a very informal description of a linked tree. A precise definition can be found in S. Peters and R.W. Ritchie, *Phrase Linking Grammars,* Technical Report, Department of Linguistics, University of Texas at Austin, 1982.

[5] S/NP NP means an S tree with two NP type holes.

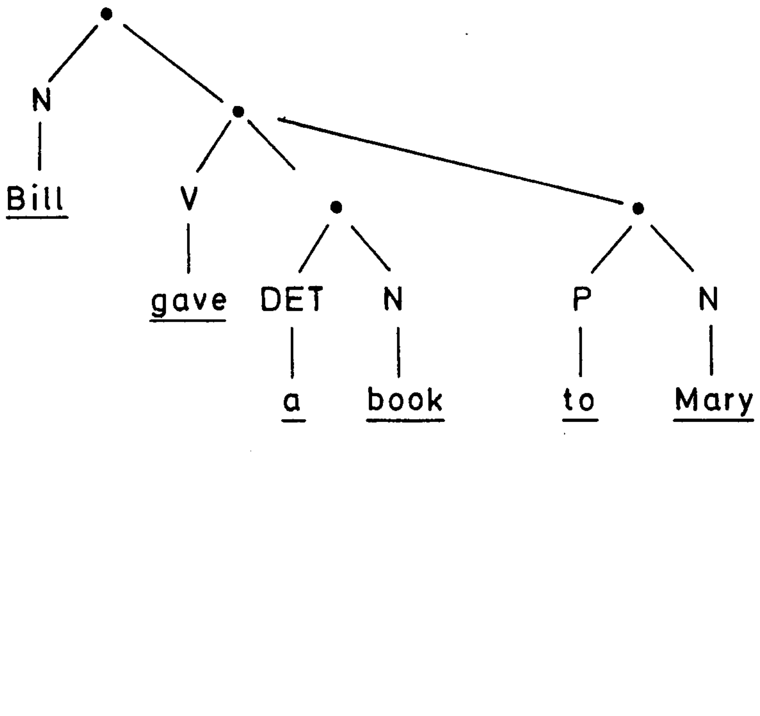
description or a *skeleton*. A skeleton exhibits all of the grouping structure without naming the syntactic categories. For example, (9) is a skeleton. The structural description is characterized only by the shape of the tree and not the associated labels. The only symbols appearing in the structure are the terminal symbols (more precisely, the preterminal symbols and the terminal symbols, in the linguistic context, as in (10); however, for the rest of the discussion, we will take skeletons to mean trees with terminal symbols only).

Let G be a context-free grammar and let T_G be the set of sentential derivation trees (structural descriptions) of G. Let S_G be the skeletons of G, i.e., all trees in T_G with the labels removed.

It is possible to show that for every context-free grammar G we can construct a skeletal generating system (consisting of skeletons and skeletal rewriting rules) that generates exactly S_G; i.e., all category labels can be eliminated while retaining the structural information (Levy and Joshi 1979).



(9)

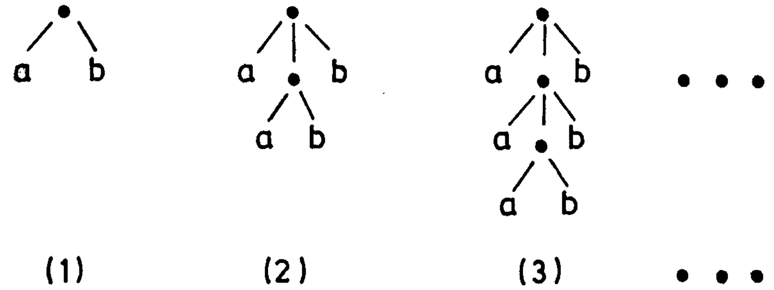


(10)

Example 6.1

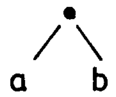
G : S → aSb
S → ab

S_G:

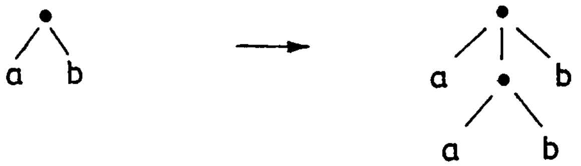


A skeletal generating system can be constructed as follows. We have a finite set of initial skeletons and a finite set of skeletal rewriting rules whose left-hand and right-hand sides are skeletons.

Initial Skeletons



Skeletal rewriting rules



In this system, generation proceeds from an initial skeleton through a sequence of intermediate skeletons to the desired skeleton. Clearly, because of the definition of a skeleton and the nature of the skeletal rewriting rules, the rules must always apply to one of the lowermost configurations in a skeleton that matches with the left-hand side of a rule. Thus the derivation of the skeleton (3) in S_G would be as in (11). The configurations encircled by a dotted line are the ones to which the skeletal rule is applied.

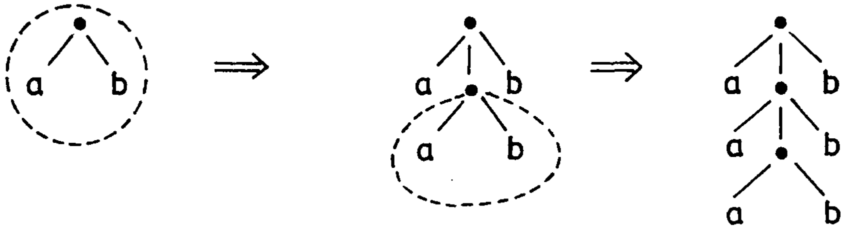
In the above example, there was only one nonterminal; hence the result is obvious. Following is a somewhat more complicated example.

Example 6.2

G : E → a
E → (E)
E → E + E/¬(+__)∧¬(*__)∧¬(__*)
E → E * E/¬(+__)

G is a local constraints grammar. Clearly there is a context-free grammar G′ that is equivalent to G. Rather than taking a complicated context-free grammar and then exhibiting the equivalent skeletal grammar, we will take the local constraints grammar G and exhibit a skeletal grammar equivalent to G. This will allow us to present a complicated example without making the resulting skeletal grammar too unwieldy. Also, this example will give some idea about the relationship between local constraints grammars and skeletal grammars; in particular, the skeletal rewriting rules indirectly encode the local constraints in the rules in Example 6.2.

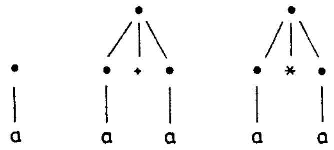
We have eliminated all labels by introducing structural rewriting rules and defining the derivation as proceeding from skeleton to skeleton rather than from string to string. This result clearly brings out the relationship between the grouping structure and the syntactic categories labeling the nodes.



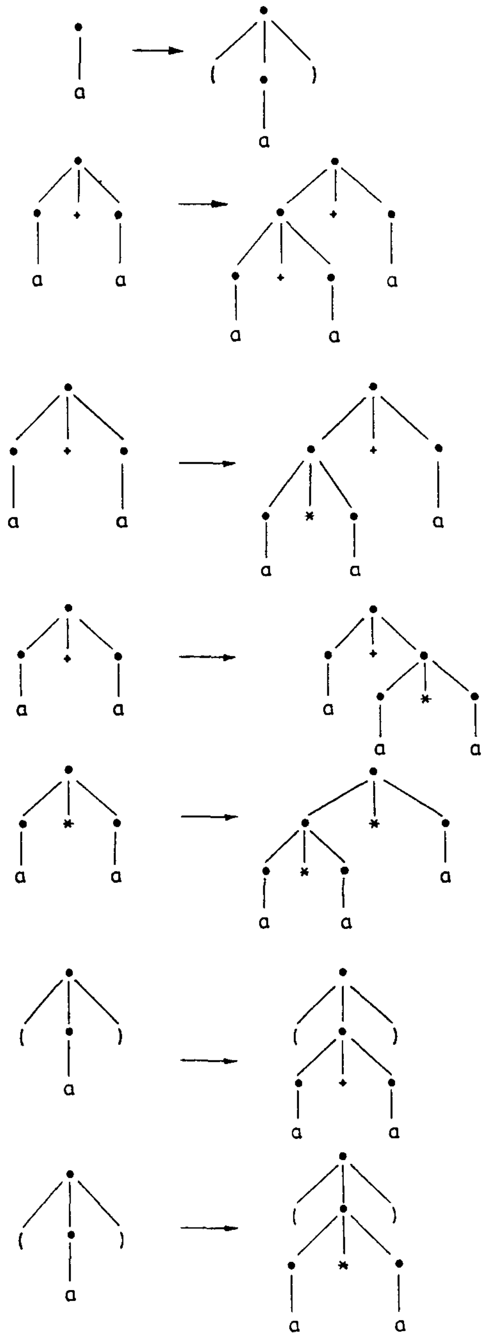
(11)

Skeletal grammar equivalent to G:

Initial Skeletons:



Skeletal Rewriting Rules:



Since skeletons pay attention to grouping only, this result may be psycholinguistically important because our first intuition about the structure of a sentence is more likely to be in terms of the grouping structure and not in terms of the corresponding syntactic categories, especially those beyond the preterminal categories.

The theory of skeletons may also provide some insight into the problem of grammatical inference. For a finite state string automaton, it is well know that if the number of states is 2k then, if we are presented with all acceptable stings of length ≤2k, the finite state automaton is completely determined. We have a similar situation with the skeletons. First, it can be shown that for each skeletal set S_G (i.e., the set of skeletons of a context-free grammar) we can construct a bottom-up tree automaton that recognizes precisely S_G (Levy and Joshi 1978). Further, if the number of states of this automaton is k, then the set of all acceptable sets of skeletons of depth ≤2k completely determines S_G (Levy and Joshi 1979). Using skeletons (i.e., string with their grouping structure) rather than just strings as input to a grammatical inference machine is an idea worth pursuing further.

6. Conclusion:

We have presented several results concerning phrase structure trees that show that phrase structure trees when viewed in certain ways have much more descriptive power than one would have thought. We have given a brief account of our work on local constraints and presented an intuitive proof. We have also compared it to some aspects of the framework of Gazdar and that of Peters and Karttunen. We have also shown that phrase structure trees, even when deprived of the labels, retain in a certain sense all the structural information. This result has implications for grammatical inference procedures.

References

Bresnan, J.W. 1978 A Realistic transformational grammar. In Halle, M., Bresnan, J. and Miller, G.A., Ed., *Linguistic Theory and Psychological Reality.* The MIT Press, Cambridge, Mass.

Chomsky, N. 1965 *Aspects of the Theory of Syntax,* The MIT Press, Cambridge, Mass.

Gazdar, G.J.M. 1978 English as a context-free language. unpublished manuscript.

Gazdar, G.J.M. 1981 Unbounded dependencies and coordinate structure. *Linguistic Inquiry* **12,** 2.

Gazdar. G.J.M. 1982 Phrase Structure Grammar. To appear in Jacobson, P. and Pullam, G.K., Ed., *The Nature of Syntactic Representation.* Reidel, Boston, Mass.

Harris, Z.S. 1956 String analysing of language structure. manuscript. Also in manuscripts around 1958 and published in 1962 by Mouton and Co., The Hague.

Joshi, A.K. and Levy, L.S. 1977 Constraints on structural descriptions. *SIAM Journal of Computing.*

Joshi, A.K., Levy, L.S., and Yueh, K. 1980 Local constraints in the syntax and semantics of programming language. *Journal of Theoretical Computer Science.*

Kaplan, R. and Bresnan, J.W. 1979 A formal system for grammatical representation. To appear in Bresnan, J.W., Ed., *The Mental Representation of Grammatical Relations.* The MIT Press, Mass.

Karttunen, L. 1980 Unbounded dependencies in phrase structure grammar: slash categories versus dotted lines. Paper presented at the Third Amsterdam Colloquium: *Formal Methods in the Study of Language,* Amsterdam.

Levy, L.S. and Joshi, A.K. 1978 Skeletal structural descriptions. *Information and Control.*

McCawley, J.D. 1967 Concerning the base component of a transformational grammar. *Foundations of Language,* Vol. 4. Reprinted in McCawley, J.D., *Grammar and Meaning,* Academic Press, New York, NY, 1968.

Peters, S. 1980 (Talk presented at the Workshop on Alternatives to Transformation Grammars, Stanford University, January.)

Peters, S. and Ritchie, R.W. 1969 Context-sensitive immediate constituent analysis. *Proc. ACM Symposium on Theory of Computing.*

Sager, N. 1967 Syntactic analysis of natural languages. *Advances in Computers, Vol. 8* ed. M. Alt and M. Rubinoff, Academic Press, New York, NY.

Aravind K. Joshi is a professor of computer and information science, and that department's chairman, at the University of Pennsylvania, Philadelphia. He received the Ph.D. degree in electrical engineering from the University of Pennsylvania in 1960.

Leon S. Levy is a member of the technical staff of Bell Telephone Laboratories at Whippany, New Jersey. He received the Ph.D. degree in Computer Science from the University of Pennsylvania in 1970.