Dependency Grammars and Context-Free Grammars

Steven Abney
University of Tübingen

The question arises from time to time what the relation is between dependency grammars (DG's) and phrase-structure grammars. A classic paper by Gaifman [1] would appear to have laid the issue to rest, by proving that dependency grammars are a special case of context-free grammars (CFG's). Gaifman proves that dependency grammars are equivalent to a proper subset of phrase-structure grammars, those of degree ≤ 1, which I will dub d1-CFG's. (As *degree* cannot be explained in a few words, I will leave it undefined for the moment.) Under a weaker notion of correspondence, which Gaifman attributes to Hays [2], dependency grammars correspond to finite-degree CFG's, which represent a larger subset of CFG's, but a proper subset nonetheless.

I submit, however, that Gaifman correlates DG's with a proper subset of CFG's only by suppressing the essential property of DG's, namely, their headedness. I would like to show that, if we take headedness seriously, we are led to the conclusion that DG's and CFG's both represent equivalence classes of what I will call *headed context-free grammars* (HCFG's), but that neither DG's nor CFG's subsume the other. Nonetheless, Gaifman's result is preserved in a different form, in that the equivalence classes defined by CFG's include all HCFG's, but the equivalence classes defined by DG's include only finite-degree HCFG's.

In particular, each HCFG has a unique *characteristic grammar*, a CFG that abstracts away from the choice of heads in the HCFG. Each HCFG also has a unique *projection grammar*, a DG representing the dependencies among *projections* in the headed trees generated by the HCFG. The projection grammar abstracts away from the order in which dependent projections are combined with governing projections. Each relation defines equivalence classes: the class of HCFG's having the same characteristic grammar, the class of HCFG's having the same projection grammar. But the equivalence classes are incomparable. There are HCFG's that have the same characteristic grammar, but different projection grammars; and there are HCFG's with the same projection grammar, but different characteristic grammars.

To flesh out this sketch, we need to define some terms. A DG is a tuple $G = (\Sigma, P, S)$, where Σ is a set of word categories, P is a set of productions, and $S \subseteq \Sigma$ is the set of start symbols. Productions are of the form $X(\alpha;\beta)$, where X is a category, and α, β are sequences of categories. The productions license dependency trees. A dependency tree is licensed by G iff every node is licensed by some production of G. A production $X(Y_1, \ldots, Y_m; Z_1, \ldots Z_n)$

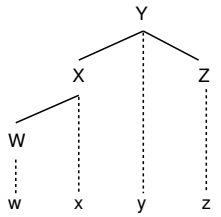
Draft of 6 March 1994

licenses a node iff the node is of category X, its left dependents are of categories $Y_1, \ldots, Y_m$, in that order, and its right dependents of categories $Z_1, \ldots, Z_n$, in that order.

To relate a dependency tree to a sequence of words, we also require a lexicon, which assigns words to categories. For example, the following DG-cum-lexicon licenses (*generates*) the dependency tree (1), among others.

$$\begin{array}{lll} X(W;\epsilon) & x \in X & S = Y \\ Y(X;Z) & y \in Y & \\ & z \in Z & \\ & w \in W & \end{array}$$

(1)

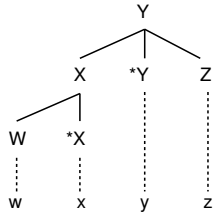


A CFG is a tuple $G = (V, \Sigma, P, S)$, where V is a set of nonterminal categories, Σ a disjoint set of terminal categories, P a set of productions, and $S \in V$ is the start symbol. The productions are of form $X \to \alpha$, where $X \in V$ and $\alpha \in (V \cup \Sigma)*$. The productions license phrase-structure trees. A terminal node must be labelled with a terminal category. A nonterminal node must be labelled with a nonterminal category, and licensed by some production in P. The production $X \to Y_1, \ldots, Y_n$ licenses a node iff the node is of category X and its children are of categories $Y_1, \ldots, Y_n$, in that order. Following Gaifman, we assume a separate lexicon that assigns words to terminal categories.

As an example, the following grammar-cum-lexicon licenses the phrase-structure tree given in (2).

$$\begin{array}{ll} X \to W\ {*X} & x \in {*X} \\ Y \to X\ {*Y}\ Z & y \in {*Y} \\ & z \in Z \\ & w \in W \end{array}$$

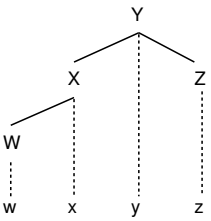
(2)

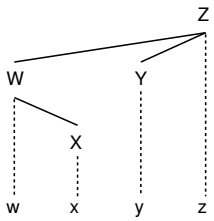


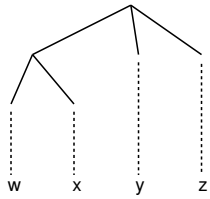
To compare dependency grammars to context-free grammars, Gaifman defines the phrase-structure tree *induced* by a given dependency tree. The induced

phrase-structure tree is identical to the dependency tree, with the exception that extra terminal nodes are introduced for internal nodes in the dependency tree. This is necessary because only terminal nodes correspond to words in a phrase-structure tree, whereas in a dependency tree, no distinction is made between terminal and nonterminal nodes. (2) is the tree induced by (1); $*X$ and $*Y$ are new categories representing the heads of X and Y, respectively.

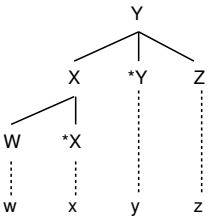
This seems innocuous enough, but it has serious consequences. Gaifman proceeds to define two grammars to be equivalent if they generate/induce the same set of *unlabelled* trees. But as Gaifman himself points out, dependency trees that differ not only in labels, but also in structure, induce the same unlabelled phrase-structure trees. For example, the dependency trees (3a) and (3b) induce the same unlabelled phrase-structure tree (3c).

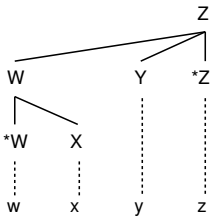
(3) a. 

b. 

c. 

Gaifman justifies his decision by pointing out that different dependency trees induce different *labelled* phrase-structure trees. For example, the dependency trees in (3) induce the labelled trees in (4).

(4) a. 

b. 

However, the results about equivalence of dependency grammars and d1-CFG's are based on unlabelled trees. Gaifman does not prove comparable results for labelled trees, and indeed, problems arise if one attempts to do so. The essential difficulty is the following. In a tree like (4a), we as readers can tell that $*X$ is intended to be the head of X, because of the star convention, but that information is not represented anywhere in the CFG. W and $*X$ are simply two different categories, and both distinct from X. So it is easy enough to construct a CFG from a DG, but it is rather more difficult to set up the other half of the equivalence relation, from a CFG back to a unique DG. It can be done, but it involves encoding a convention for marking heads (like the star convention) into the definition of the equivalence relation.

As a result, the claim that DG's are equivalent to d1-CFG's loses much of its interest. After all, since DG's and d1-CFG's are both countably infinite sets

(assuming fixed, infinite sets of categories), we know that equivalence relations between them exist, and it's not too surprising if we can construct one, with a bit of cleverness. But we could equally well construct an equivalence relation to show that the class of CFG's is equivalent to some proper subset of DG's. For example, we could convert each CFG to Greibach normal form, following a convention regarding category names to guarantee that each CFG is mapped to a different GNF grammar, then map the GNF grammar to a DG in the obvious way. This construction would permit us to say that CFG's are equivalent to a proper subset of the DG's, namely, those in which no production has any left-dependents.

In brief, mapping dependency trees to phrase-structure trees is interesting only if no significant properties of the dependency trees are suppressed. Since the distinction between a node and its dependents is clearly a significant property of dependency trees, we need to preserve it in the mapping.

Therefore, instead of mapping dependency trees to phrase-structure trees *simpliciter*, let us map dependency trees to *headed phrase-structure trees*. In a headed phrase-structure tree, a unique child of each node is distinguished as the head. Formally, we define a node (without label) as a pair (α, i), where α (a sequence of nodes) is the node's children, and i is the index of the head node. Notationally, Gaifman's star convention is as good as any other, so we will write headed phrase-structure trees as in (4).[1] However, the stars are now to be understood not as part of the category-name, but as a mark distinguishing the head node. The unlabelled headed tree corresponding to (4a) is not (3c), but rather, (3c) with stars on the second and third terminal nodes. The unlabelled headed tree corresponding to (4b) would have stars on the first and fourth terminal nodes instead.

To generate headed phrase-structure trees, we define a *headed context-free grammar* to be a context-free grammar in which a unique element on the right-hand side of each production is distinguished as the head. Formally, we define productions as pairs (r, i), where r is a CFG production, and i is the index of the head, with $1 \leq i \leq |r|$. (We assume there are no epsilon productions. With a separate lexicon, we can replace epsilon with an epsilon terminal category. In the lexicon, the empty string is assigned to the epsilon terminal category.)

The *characteristic grammar* $C(G)$ of a HCFG G is the CFG obtained by omitting the head indices. Every HCFG has a unique characteristic grammar, and every CFG is the characteristic grammar of at least one HCFG. C is not one-one. More than one HCFG may have the same characteristic grammar; or said the other way round, we may choose heads for a CFG in more than one way.

Two HCFG's are characteristic-equivalent if they have the same characteristic grammar. Two characteristic-equivalent grammars are also strongly

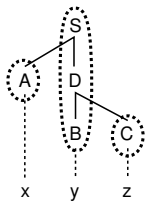
[1] Another convenient convention, which I will adopt without further comment, is to draw trees such that heads are connected to their parent by a vertical line, and non-heads are connected to their parent by oblique lines.

equivalent, in the following sense. We can map headed phrase-structure trees to unheaded phrase-structure trees by erasing the head-markings (informally speaking). In this manner, we can define the unheaded tree-set generated by an HCFG by mapping each headed tree it generates to an unheaded one. An HCFG is strongly equivalent to its characteristic grammar in the sense that both generate the same set of unheaded trees.

We can also relate an HCFG to a unique DG, which I will call its *projection-dependency grammar*. A *projection* in a headed tree is a path from a terminal node upward, from head to parent, terminating in a node that is not the head of its parent. The *degree* of a headed tree is the length of the longest projection it contains. We define the degree of a headed grammar as the maximal degree of any headed tree it generates. The degree of a CFG, in turn, is the *minimal* degree for any choice of heads (yielding an HCFG). This is not Gaifman's definition of degree, inasmuch as he does not talk about headed trees or headed grammars, but it is extensionally equivalent as applied to CFG's.

We next need to assign categories to projections. Define a *projection-category* to be a sequence of categories. A projection is a sequence of nodes; its projection-category is the corresponding sequence of node categories. For example, there are three projections in tree (5), and their projection-categories are (A), (B, D, S), and (C), respectively.

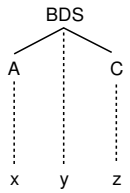
(5)



We can also define dependencies between projections. In a headed tree T, we say that a projection Q *depends on* a projection P just in case the root of Q is child of some node in P. The *dependents* of a projection P are those projections that depend on it. P's left dependents are those that precede it, and its right dependents are those that follow it. For example, in (5), the projection (B, D, S) has one left dependent, (A), and one right-dependent, (C).

With these definitions, we can map a headed tree to a unique dependency tree, its *projection-dependency tree*. Each projection P in the headed tree defines a node π in the projection-dependency tree. The category of π is the projection-category of P. The projections $Q_1, \ldots, Q_n$ that are dependent on P in the headed tree map to the dependents of π in the projection-tree. For example, the projection-dependency tree of (5) is (6):

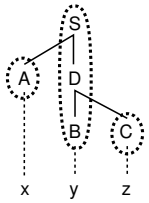
(6)



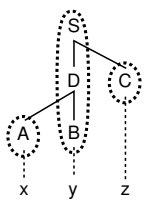
We say that an HCFG generates a projection-dependency tree τ just in case it generates some headed tree whose projection-dependency tree is τ.

The projection-dependency tree of a headed tree is unique, but more than one headed tree may have the same projection-dependency tree. Informally speaking, projection-dependency trees abstract away from the order in which dependents are combined with their governor. For example, (6) is the projection-dependency tree of both (7a) and (7b):

(7) a.



b.



The definition of projection-dependency trees permits us to define the dependency grammar corresponding to a headed context-free grammar. The *projection grammar* $\Pi(G)$ of an HCFG G is a dependency grammar, defined as follows. The categories of $\Pi(G)$ are the projection-categories of G. The start symbols of $\Pi(G)$ are all projection-categories that end with the start symbol of G. The productions of $\Pi(G)$ include $X(\alpha;\beta)$ just in case there is a node with category X and with dependents of categories α, β in some projection-dependency tree generated by G.

The projection grammar of an HCFG is well-defined just in case the defined sets of categories and productions are finite. The set of projection-categories and the set of dependency productions are finite iff there is a bound on the length of projections, that is, iff the HCFG is of finite degree.

Every finite-degree HCFG has a unique projection grammar, and every DG is the projection grammar of at least one HCFG.[2] But more than one HCFG may have the same projection grammar. From the other perspective, there is more than one way in which to combine dependents with their governor, producing a projection. Two HCFG's are projection-equivalent if they have the same projection grammar.

It is straightforward to show that an HCFG and its projection grammar generate the same set of (projection-)dependency trees. It follows that, if two

[2]To be quite precise, every DG is *isomorphic* to the projection grammar of at least one HCFG. Not all DG's use projection-categories as categories; but every DG is identical up to choice of category names to some DG that is a projection grammar.

HCFG's are projection-equivalent, they are strongly equivalent in the sense of generating the same (projection-)dependency trees. (This is of course a different sense of strong equivalence from that relating HCFG's to their characteristic CFG's.)

As we have seen, both C and Π induce equivalence classes of HCFG's. But the equivalence classes defined by C and Π are incomparable. There are HCFG's that have the same characteristic grammar, but different projection grammars. Conversely, there are HCFG's with the same projection grammar, but different characteristic grammars. (8a) provides an example of the former; (8b) of the latter.

(8) a.

G	$S \to *a\ b$	$S \to a\ *b$
$C(G)$	$S \to a\ b$	$S \to a\ b$
$\Pi(G)$	$aS(\epsilon; b)$	$bS(a; \epsilon)$

b.

G	$S \to a\ *A$	$S \to *A\ c$
	$A \to *b\ c$	$A \to a\ *b$
$C(G)$	$S \to a\ A$	$S \to A\ c$
	$A \to b\ c$	$A \to a\ b$
$\Pi(G)$	$bAS(a; c)$	$bAS(a; c)$

It is at any rate not accurate to say that DG's are a special case of CFG's. It *is*, however, accurate to say that DG's are more limited than CFG's in a certain sense. Namely, the equivalence classes defined by CFG's include all the HCFG's, but the equivalence classes defined by DG's include only the finite-degree HCFG's. DG's do not permit a word to have unboundedly many dependents, but HCFG's do countenance that possibility.

In closing, it is perhaps appropriate to discuss the import of the result. From a mathematical perspective, equivalences between types of representation provide useful tools. Some results are easier to discover or prove using one type of representation; some are easier to discover or prove using the other. The equivalence allows one to use either type of representation interchangeably.

Judging from the literature, the linguist's attitude tends to be rather different. If one shows that two representations are equivalent in some respect, the immediate response is to seek other arguments for why one or the other representation is right. Either the correct structure is a phrase-structure tree, or it is a dependency tree, but it certainly cannot be both. Assigning both a phrase-structure tree and a dependency tree to the sentence would be at best absurdly redundant. Accordingly, either DG is the correct grammatical formalism, or CFG, but not both.

Examples of this sort of reasoning are abundant in the literature. For instance, in GB, a concensus crystallized about the time of Stowell's dissertation that phrase-structure rules were redundant with Case theory and θ-theory, etc.;

and since then, phrase-structure rules have been virtually taboo. Or again, in the area of GB parsing there has been a general concensus that, with respect to the structures assigned to sentences, a vector of parameter settings is probably equivalent to some augmented context-free grammar. Nonetheless, the view is adopted very persistently that the GB description is true, and the augmented context-free description is false, and the parser must therefore use as data structures GB principles and parameters.

One contributor to this attitude is perhaps Chomsky's early characterization of simplicity in terms of the number of symbols used in writing down a grammar. That view of simplicity is a valid approach in the context of grammatical inference, where the grammar to be infered is drawn from a specified class of grammars, and one must choose somehow among the grammars that all describe the input sample equally well. But Chomsky has long since abandoned the view that grammatical inference consists in choosing from the class of TG's the simplest TG compatible with the input.

A grammar is a mathematical characterization of a set of syntactic structures. Any other equivalent description is to be freely used if it is more convenient for some purpose. As for the syntactic structures, they are also mathematical descriptions of properties of sentences, and any equivalent description may be used if it is more convenient for establishing some result or stating some constraint. The ability to translate among representations is itself a powerful and desirable tool.

If the result reported here is of linguistic interest, it is because some constraints on syntactic structures are most easily established using projection-dependency trees, and others are most easily established using characteristic trees. The result does not mean that phrase-structure trees are right and dependency trees are wrong. Indeed, if one is a true description of a sentence, the other is ipso facto true; though they may be differentially useful in different contexts.

References

[1] Haim Gaifman. Dependency systems and phrase structure systems. Technical Report P-2315, The RAND Corporation, Santa Monica, CA, May 1961.

[2] D.G. Hays. Grouping and dependency theories. Research Memorandum RM-2538, The RAND Corporation, 1960.