METHODS FOR OBTAINING CORRESPONDING PHRASE STRUCTURE AND DEPENDENCY GRAMMARS

Jane J. Robinson

International Business Machines Corporation
Thomas J. Watson Research Center
Yorktown Heights, New York

ABSTRACT Two methods are given for converting grammars belonging to different systems. One converts a s̲imple (context-free) phrase structure grammar (S̲PG) into a corresponding dependency grammar (DG); the other converts a DG into a corresponding SPG. The structures assigned to a string by a source grammar will correspond systematically, though asymmetrically, to those assigned by the target grammar resulting from its conversion. Since both systems are weakly equivalent, generating exactly the CF languages, the methods facilitate experimentation with either notation in devising rules for any CF language or any CF set of strings designed to undergo subsequent transformation.

A source SPG is assumed to be of finite degree with ordered rules in which only the initial symbol is recursive. Unless the source grammars obey additional constraints, the target grammars may exhibit a peculiar property, defined as "structure sensitivity". The linguistic implications of the property are discussed, and the linguistic motivation for imposing the constraints necessary to avoid its appearance is suggested.

The author owes an especial debt to Jesse Wright of the Automata Theory and Computability Group in the Mathematical Science Department of IBM Research for many helpful discussions of theoretical problems arising in the course of this investigation.

In an article on dependency theory[1] appearing in 1964 [5], Hays remarked, "Casual examination suggests there would be little difference between transformation of dependency trees and transformation of IC [immediate constituent] structures, but no definite investigation has been undertaken." Since then, a dependency grammar with transformational rules has been designed for a subset of English sentences, and preliminary results indicate that Hays' observation is correct. [9] In either case, transformations are specified in terms of labeled trees, and the number of branches and the denotations of the labels do not affect the essential operation. Since the base grammar is generally limited to the generation of context-free pre-terminal strings of "deep structure", and since context-free languages are characterizable in either dependency or phrase structure (i.e., immediate constituent) systems, neither system is clearly preferable as a base. A linguist may find that the notation afforded by one or the other is simpler for characterizing some language, or for defining structures to be transformed, or for adapting a grammar to computer applications. A linguist may also wish to experiment with grammars of both types, or redesign transformations defined on the structures of one base grammar in order to incorporate them into a transformational grammar using a different base.

Such considerations as these motivate the present treatment of the problem of obtaining paired grammars by converting a grammar of one kind into a systematically corresponding grammar of the other which generates the same sentences and assigns comparable structures. In addition, conversion draws attention to some linguistically significant relationships that may exist unnoticed among the categories and rules of the source grammar and which may induce in the derived grammar a peculiar property of structure sensitivity, roughly analogous to context sensitivity. This property will be exhibited and discussed in the course of illustrating the method.

We begin concretely by inspecting (Fig. 1) a pair of grammars: SPG1, a simple or context-free phrase structure grammar of the kind formalized by Chomsky [2], Bar-Hillel [1], and others, and DG1, a dependency grammar of the kind formalized by Gaifman [4] and Hays [5]. Two structural diagrams, a P-tree and a D-tree drawn beneath the grammars, illustrate the structure each assigns.

The rewriting rules of SPG1 are of two kinds, those in which only categories appear, and those in which a category is rewritten as a terminal (lexical formative or word). The latter may be separated from the former and made into assignment rules,

[1] For additional material on dependency theory, see Ref. 6.

SPG1

Axiom: # S #

Rewriting Rules:

1. S → NP VP
2. VP → V NP
3. NP → D N
4. D → the
5. D → some
6. N → boys
7. N → girls
8. V → like
9. V → admire

(1a)

DG1

Axiom: * (V)

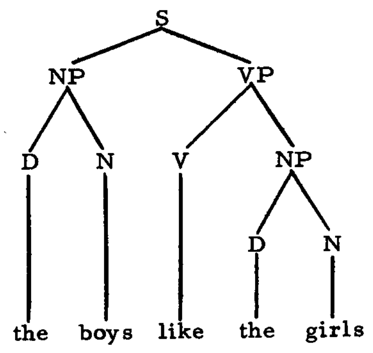
Dependency Rules:

1. V (N * N)
2. N (D *)
3. D (*)

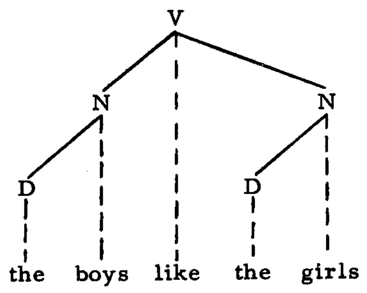
Assignment Rules:

1. D: the, some
2. N: boys, girls
3. V: like, admire

(1b)



(1c)



(1d)

Figure 1

thereby increasing the resemblance of SPG1 to DG1 in an obvious way. This is possible because SPG1 does not contain any mixed rewriting rules in which both categories and lexical formatives appear on the right. It can be shown that any SPG which has mixed rules, in this sense, can be converted into one which does not merely by introducing new categories, without affecting generative power. Thus there is no reason for not separating the two types of rules, and Chomsky devotes a good part of _Aspects of the Theory of Syntax_ [3] to giving linguistic reasons for just such a separation.

Hereafter, "rewriting rules" will refer only to those like rules 1-3 in SPG1, and those like rules 4-9 will be called "assignment rules". Categories which do not appear on the left of any rewriting rules are _terminal categories_.

With each category of SPG1, we associate a number called its degree. (To say that a category is of degree n means that n is the fixed upper limit to the number of nodes of the shortest path leading from it to a terminal category in any structure derived from it by successive rule applications. A category may be of infinite degree. For example, if $X \rightarrow XX$, then X is of infinite degree and so is the grammar in which it occurs even though another rule rewrites X as a string containing only terminal categories.)[1] In SPG1 the terminal categories D, N, V are of zero degree, since they are not rewritten; the categories NP and VP are of degree 1, since at least one category in their replacement is of zero degree; and the degree of S is 2, since the least degree assigned to any category on the right of the rule for rewriting S is 1. The maximum number assigned to any category in a grammar is also the degree of the grammar; therefore, the degree of SPG1 is 2.

An essential difference between SPG1 and DG1 now emerges more clearly. DG1 uses only terminal categories, while SPG1 uses categories of higher degree. The effects of the differences are reflected in the structure of the P-tree and D-tree. The latter has fewer branches, and this will always be the case for structures assigned to the same string by grammars belonging to the two different systems.

Even so, there is a systematic correspondence between the two trees and their labels of a kind defined by Hays [5] . Every complete subtree of the D-tree, that is, every node taken together with all of its descendents, covers a substring of the sentence that is covered by a complete subtree of the P-tree. The converse does not hold, but every complete subtree of the P-tree covers a substring that is covered by a connected subtree of the D-tree. In

[1] For a more precise characterization of "degree", see Gaifman [4].

the example, both complete subtrees dominated by N in the D-tree correspond to the two complete subtrees of the P-tree dominated by NP, and the complete (sub)tree dominated by V corresponds to the complete (sub)tree dominated by S. However, the complete subtree dominated by VP in the P-tree corresponds to an <u>incom</u>plete subtree of the D-tree, which is dominated by V but includes only V and the branches to its right; so that the relationship of correspondence is asymmetrical.

While such systematic correspondences exist between the structures assigned to all strings generated by SPG1 and DG1, this is not the general case. In general:[1]

1. Any context-free language can be generated by grammars of either simple phrase structure or dependency systems.
2. For any given SPG, there exists one or more DG over the same language all of whose structures correspond systematically to structures assigned to the same strings by the SPG if and only if the SPG is of finite degree.
3. For any given DG there exists one or more corresponding SPG.
4. For any given DG there exists a unique SPG of degree 1 that is strongly equivalent to it.

Gaifman [4] gives a very general method for constructing corresponding DG from any SPG of finite degree, and also a method for constructing a unique SPG of degree 1 from any DG. Here we give two methods for constructing corresponding SPG and DG that differ from Gaifman's. The first applies only to a more restricted set of SPG and leads to reduced DG, whereas Gaifman's method tends to produce DG with overlapping categories and superfluous rules that may never be used to generate any string. The second allows construction of SPG of degree greater than 1 from certain DG.[2]

A simplified sketch of the two methods follows. Each rewriting rule is "augmented" by starring a category on the right. Each dependency rule is augmented by assigning a numerical coefficient to each dependent. Figure 2 shows a possible augmented

[1] Proofs are given by Gaifman [4]. It is assumed that the SPG is non-erasing and reduced; that is, no category is rewritten as null, and there are no superfluous categories or rules.

[2] The second method was suggested by Kay's procedure for constructing P-trees from D-trees [7]. More precisely, an SPG of degree $n > 1$ may be derived if, in the DG, some categories govern two or more dependents, and the left- or rightmost dependent itself governs dependents.

form of SPG1 on the left and a possible augmented form of DG1 on the right. (Detailed consideration of the problem of augmentation will follow the sketch of the operations.) These augmentations were deliberately chosen so that conversion of either grammar uniquely produces the other. Different augmentations produce different results, although structures of the original grammar and of the grammar derived from it still correspond.

SPG1	TS	IRL	DG1
$S \rightarrow NP\ VP*$	$S{:}V^2$	$V^2 \rightarrow N^1\ V^1_*$	V (2N * 1N)
$VP \rightarrow V*\ NP$	$VP{:}V^1$	$V^1 \rightarrow V^0_*\ N^1$	N (1D *)
$NP \rightarrow D\ N*$	$NP{:}N^1$	$N^1 \rightarrow D^0\ N^0_*$	D (*)

Figure 2

The columns TS and IRL in Figure 2 represent a Table of Substitutes and an Intermediate Rule List. In the conversion of SPG1, the TS is constructed, equating each category on the left of a rule with a superscripted terminal category. The numerical superscript (hereafter called the exponent) equals the number of rules traced through when tracing by starred categories before a starred terminal category is reached, and expresses the distance between the terminal category and the category for which it is a substitute. In the IRL, the categories occurring in each rule of the SPG are replaced by their substitutes from TS. Taking the first IRL rule, construction of a dependency rule is begun by replacing the arrow with parentheses enclosing the substitute categories on the right. Thus from the first IRL rule in Figure 2 we obtain $V^2\ (N^1\ V^1_*)$. The construction of a dependency rule is continued so long as the exponent of the starred category X^s_* is greater than zero. The next step is to search the IRL for a rule with X^s on the left. The categories on the right of the new rule are inserted in the parentheses of the D rule under construction, in the position of X^s_*, but no new parentheses are added. When the starred occurrence is X^0_*, it is replaced by *, all exponents are erased, and the dependency rule is complete. The process is repeated for new dependency rules until all IRL rules are exhausted.

We must add rules to the constructed DG for any unstarred terminal category of the IRL. The added rules assign no dependents and are of the form D(*). The assignment rules are simply transferred, and since V is the substitute for the axiom category S, it is taken as the axiom for the DG.

Going in the other direction from an augmented DG1 to

SPG1, we first assign an exponent s to each category, where s equals the largest coefficient of any dependent of that category. If a category is not assigned dependents, its exponent is zero. The first rule of augmented DG1 now appears as V^2 ($2N^1$ * $1N^1$). From this rule two rules are constructed for the IRL; that is, the number of IRL rules constructed from each rule of DG1 will be equal to the exponent of the governor. The first rule so obtained writes the governor with its exponent on the left of the arrow. All of its dependents whose coefficients equal that of the exponent are written in order on the right, and the governing category with its exponent decreased by 1 is written with the *. Thus we obtain $V^2 \rightarrow N^1\ V^1_*$. The second rule is obtained in a similar fashion, with the exponent of the governing category diminished by 1 yielding $V^1 \rightarrow V^0_*\ N^1$. The process is repeated until an IRL rule rewriting some category with exponent equal to 1 is used, after which the next DG rule is processed, and so on until all DG rules are exhausted.

At this point, the *'s may be erased and, except for category labels, the IRL is exactly equivalent to the rewriting rules of the original unaugmented SPG. The only function the TS serves is to reassign labels. Assignment rules are transferred and the substitute for the axiom of DG1 is added.[1]

SPG1 and DG1 are very simple, with no embeddings and no optional rules. More complicated grammars give rise to problems of augmentation, especially for SPG. Even SPG1 poses a problem. Assume it had been augmented by starring NP in rule 1 and in rule 2. In that case, the same substitute, N^2, is assigned to both S and VP, and the procedure produces a DG that is not even weakly equivalent to the original. Clearly some constraints must be imposed on augmentation and provision made for grammars in which it may not be possible to avoid starring a category more than once.

Similarly, assume that some DG has a rule of the form C (A B * D), and that this is augmented as C (1A 2B * 1D). The IRL rules are:

$$C^2 \rightarrow B\ C^1_*$$
$$C^1 \rightarrow A\ C^0_*\ D.$$

But now the rules generate the sequence B A C D, which was not generated by the original rule, and do not generate A B C D, which was. This is remedied by requiring that if a coefficient n has been assigned to a dependent, no higher number is assigned to any dependent which intervenes between it and the *. We will also require that at least one dependent be preceded by 1, and if any dependent is preceded by $n > 1$, there must be at least one dependent

[1]Although grammars with only one axiom are illustrated, grammars with more than one axiom can obviously be handled as well.

preceded by n - 1. This is not crucial, but it avoids setting up unnecessary, single-branching categories in the derived SPG. Note that if all dependents in any rule are preceded by 1, which is the same as not augmenting the DG at all, the resulting SPG will be of degree 1; that is, each rule of the grammar will contain at least one terminal category on the right. This is equivalent to Gaifman's procedure [4].

Augmentation of SPG is the more difficult case. Primarily it is the problem of constructing the TS in a finite number of steps. For example, if for S $\rightarrow$ NP VP S the S is starred on the right, an infinite loop is created immediately. This is easy to avoid when considering a single rule or a small set of rules, but we do not know in general whether some series of rule augmentations may not lead to the same situation. Gaifman's solution is to require that the marked category be of lesser degree than the category on the left, but this not only leads to a proliferation of categories in the derived DG whenever the SPG has more than one rewriting rule for any of its categories, it prevents us from deriving the simplest corresponding DG in some cases.[1] On the other hand, the method employed here will not work unless some restrictions are imposed on augmentation which also imply restrictions on the form of the SPG in addition to the requirement of finite degree. It is not clear what restrictions are minimally necessary, but it is sufficient to require, in addition to finiteness of degree, that the rules of the SPG be ordered, so that in a developing derivation, if rule n has been applied, no rule m, $m < n$, need be applied thereafter. This is too restrictive, and does not allow for full recursion. We may, however, allow a dummy symbol, S', to appear in any rule rewriting some X as a string containing some Y, $S' \neq Y \neq X$, where S' is replaced after one application of the set of rules by # S #, the axiom, so that the rules then reapply in linear order.

Any SPG of finite degree with ordered rules providing for recursive application in the manner specified can be converted to a corresponding DG by the method given here. The base component proposed by Chomsky [3] for transformational grammar is an SPG of this form.[2]

[1] "Simplest" with respect to process, number of categories, and freedom from the property of structure sensitivity. See p. 14 ff.

[2] Chomsky does not explicitly require that if S' appears in rewriting X, some Y ($S' \neq Y \neq X$) appear in the rule also, but implicitly observes the restriction. A new grammar for English by Rosenbaum [10] contains a rule NP $\rightarrow$ NP S' which does not observe it. This is the only case known to me of an actual grammar for some "real" language that violates this restriction, and Rosenbaum does not claim any strong theoretical motivation for the rule in question. Cf. also Lees [8], and the MITRE grammar [11].

We turn now to more detailed consideration of augmenting and converting SPG subject to the above constraints. While the constraints insure that the SPG can be workably converted, some linguistically significant problems arise if we consider how to derive a simple DG, and it will be shown that under some conditions the derived DG will exhibit a feature not heretofore considered in the literature, a feature of "structure sensitivity".

Method 1. Conversion of SPG to DG

Step 1. Augmentation

All rules of the SPG which rewrite a given category are conflated and written as one schematized rule, with square brackets enclosing optionally omitted categories and braces enclosing lists of categories from which a single choice is made.[1] Thus $Z \rightarrow [W] \begin{Bmatrix} X \\ Y \end{Bmatrix}$ conflates four rules, and will, in any given application, rewrite Z as either WX or WY or X or Y.

Beginning with the first rule rewriting some X and proceeding in rule order, star occurrences of categories, excluding the dummy symbol S', on the right in such a way that one and only one starred category Y*, where $Y \neq X$, will occur in any application of the rule.[2] It follows that no bracketed (omissible) occurrence may be starred. If more than one category is starred, the schematized rule must be separated into as many rules as there are starred categories, with one starred category appearing in each. For example,

$X \rightarrow \begin{Bmatrix} A^* & \begin{Bmatrix} B \\ C \end{Bmatrix} \\ D^* & F \end{Bmatrix}$ must be separated into $X \rightarrow A^* \begin{Bmatrix} B \\ C \end{Bmatrix}$ and

$X \rightarrow D^*\ F$. This is a partial undoing of the conflation, but for linguistic reasons all ways of rewriting a category will usually have some element in common and some conflation will usually be preserved.

[1] Square brackets are used here rather than the customary parentheses to avoid confusion with the parentheses used in the derived DG. More exactly, [ ] and { } enclose lists of strings of categories. In the case of [ ] the list may consist of one member, and in both cases the strings may consist of one item.

[2] Since the SPG must be finite, no rule will rewrite X as a string of X's in any application, and some other category of degree less than X will be available for starring in all cases.

If the simplest DG corresponding to the SPG is desired, then it is preferable to augment the SPG in such a way that

a) no category occurs both starred and non-starred on the right, and

b) no category is starred on the right of two or more rules which rewrite different categories.

But it is not always possible to observe these policies, and in that case, additional augmentation is necessary in order to distinguish among occurrences of X as the marked constituent in the rule rewriting Y, X as the marked constituent in the rule rewriting Z, $Z \neq Y$, and X as an unmarked constituent of any category.

Assume some category X is starred in two or more rules which rewrite different categories,[1] say Y and Z, and also occurs unstarred on the right in some rules. The two X*'s are assigned different subscripts to avoid assigning the same substitutes for Y and Z with the consequent loss of essential information. We now have three varieties of X, namely: X, X_1, and X_2, where X is the unstarred variety, X_1 is the marked constituent of Y, and X_2 of Z. (If X does not occur unstarred, only X and X_1 are needed.) If X is not a terminal category, there is a rule rewriting X. Then we must add, beneath that rule, rules for rewriting X_1 and X_2. If $X \rightarrow U^* \ W$, we add $X_1 \rightarrow U_1{}^* \ W$

and $X_2 \rightarrow U_2{}^* \ W$.

Now, if U is not a terminal category, we add rules for rewriting U_1 and U_2 beneath the rule rewriting U. Thus the process of adding rules is iterative, but it will eventually end when, in some lower rule, a terminal category is starred.

Note that in some cases sub-subscripts are needed. For example:

1. $Z \rightarrow \ldots P_1{}^* \ldots$
2. $X \rightarrow \ldots R_1{}^* \ldots$
3. $Y \rightarrow \ldots R_2{}^* \ldots$
4. $R \rightarrow \ldots P_2{}^* \ldots$
5. $P \rightarrow \ldots A^* \ldots$

---

[1] X* may occur more than once in rules rewriting the same category, Y.
E. g., in $Y \rightarrow \begin{Bmatrix} X & Y \\ Y & X \\ Z & \end{Bmatrix}$, the only possible starring is

$Y \rightarrow \begin{Bmatrix} X^* & Y \\ Y & X^* \\ Z^* & \end{Bmatrix}$, which produces three rules for Y, in two of which X* occurs.

Proceeding down the rules, we see first that P_1* occurs, and scanning down the left, that P's are not terminal categories and that there is no rule rewriting P_1. Therefore,

beneath $P \rightarrow \ldots A* \ldots$

we add $P_1 \rightarrow \ldots A_1* \ldots$.

In the second rule, R_1* occurs, is non-terminal, and requires a rule, so

beneath $R \rightarrow \ldots P_2* \ldots$

we add $R_1 \rightarrow \ldots P_{2_1}* \ldots$.

R_2* occurs in the third rule, under the same conditions, and

beneath $R_1 \rightarrow \ldots P_{2_1}* \ldots$

we add $R_2 \rightarrow \ldots P_{2_2}* \ldots$.

P_2* occurs in the fourth rule, under the same conditions, and

beneath $P_1 \rightarrow \ldots A_1* \ldots$

we add $P_2 \rightarrow \ldots A_2* \ldots$

Our rules are now:

1. $Z \rightarrow \ldots P_1* \ldots$
2. $X \rightarrow \ldots R_1* \ldots$
3. $Y \rightarrow \ldots R_2* \ldots$
4. $R \rightarrow \ldots P_2* \ldots$
5. $R_1 \rightarrow \ldots P_{2_1}* \ldots$
6. $R_2 \rightarrow \ldots P_{2_2}* \ldots$
7. $P \rightarrow \ldots A* \ldots$
8. $P_1 \rightarrow \ldots A_1* \ldots$
9. $P_2 \rightarrow \ldots A_2* \ldots$.

and we are looking at rule 5 where $P_{2_1}*$ requires us to add

10. $P_{2_1} \rightarrow \ldots A_{2_1}* \ldots$.

Rule 6 requires us to add

11. $P_{2_2} \rightarrow \ldots A_{2_2}* \ldots$.

But in rule 7, $A*$ requires no additions, because A is not subscripted, and the subscripted A's in rules 8-11 require no additions because A's are terminal categories.

As a result, no X_i* occurs on the right of two or more rules which rewrite different categories.

This process and the remaining steps will be illustrated using SPG2. In order to show the effects of different choices in augmentation, SPG2 will be augmented in a way that deliberately violates the policies advocated for starring (but not the restrictions).

SPG2

Axiom: S

Rewriting Rules:

1. $S \rightarrow NP^* \ VP$
2. $VP \rightarrow V \ NP^* \ [NP]$
3. $NP \rightarrow D \ N^*$

We will assume assignments are the same as for SPG1 with the additional assignments of "send" and "give" to V, and "books" and "flowers" to N. [1]

Note that, in this augmentation, NP is starred twice and also occurs non-starred on the right. Subscripting and duplication are required, resulting in

1. $S \rightarrow NP^*_1 \ VP$
2. $VP \rightarrow V \ NP^*_2 \ [NP]$
3. $NP \rightarrow D \ N^*$
4. $NP_1 \rightarrow D \ N^*_1$
5. $NP_2 \rightarrow D \ N^*_2$

Step 2. Establish a table of substitutes (TS) of 2 columns and n rows, where n equals the number of rules in the SPG (after step 1). List categories on the left in column 1, and starred categories on the right in column 2, in order. Eliminate any duplicate rows. (Cf. p. 9, footnote 1.)

At the end of step 2, applied to SPG2, the result is

TS

S	NP_1
VP	NP_2
NP	N
NP_1	N_1
NP_2	N_2

[1] In a transformational grammar, lexical assignment rules of more complex form than that given here would presumably block the generation of un-English sentences.

Step 3. Starting with k = 1, try to match the category in row k, column 2 (k, 2) with a category occurring in column 1 of a lower row. If a match is found on row m, check to see if a match also occurs on m + 1. (This will be the case if the SPG contains more than one rewriting rule for the category in (k, 2).) If it does, mark m as a branching point, insert a duplicate of row k beneath row k (the duplicate will be row k + 1) and follow separate branches for substitutes for (k, 1) and (k + 1, 1). Replace the category in (k, 2) with the category in (m, 2) and repeat the search on the remaining lower rows. When the search is exhausted, assign an exponent s to the last category obtained (the final substitute) in column 2, where s equals the number of matches plus 1. Increment k and repeat until every category in column 1 has a substitute that does not appear in column 1. At the end of step 3, we obtain a unique substitute for every rewritten category of SPG2.[1]

TS

S	N_1^2
VP	N_2^2
NP	N^1
NP_1	N_1^1
NP_2	N_2^1

Step 4. Construct an Intermediate Rule List (IRL) by replacing each category in the augmented SPG rules with its substitutes from TS. If a category has no substitute, superscript it with a zero. If X_i has more than one substitute, include them in braces wherever X_i appears on the right. If a category X occurring on the left has more than one substitute, provide a separate rewriting rule for each substitute Y_n^s such that $Y_n^s \rightarrow \ldots Y_n^{s-1}* \ldots$.[2] If a non-starred category on the right

[1]In cases, not illustrated here, where several substitutes are found because several rules rewrite some X_i, each substitute will be uniquely assigned to X_i.

[2]E.g., assume
$X \rightarrow \ldots A* \ldots$
$X \rightarrow \ldots Y* \ldots$
$Y \rightarrow \ldots B* \ldots$
$Y \rightarrow \ldots C* \ldots$
so that the substitutes for X are A^1, B^2, and C^2, and the substitutes for Y are B^1 and C^1. Then the IRL is:

has more than one substitute, include all substitutes as braced options. At the end of step 4, we obtain

1. $N_1^2 \rightarrow N_1^1* \; N_2^2$
2. $N_2^2 \rightarrow V^0 \; N_2^1* \; [N^1]$
3. $N^1 \rightarrow D^0 \; N_*^0$
4. $N_1^1 \rightarrow D^0 \; N_1^0*$
5. $N_2^1 \rightarrow D^0 \; N_2^0*$

Step 5. Take the first unmarked IRL rule, which rewrites some X_n^s, set s as a counter, and write X_n followed by a pair of parentheses enclosing the string of categories on the right of the IRL rule. Note that the string will contain an $X_n^{s-1}*$.

Step 6. Mark the previously processed IRL rule, decrease s by 1, and test for s = 0. If so, go to step 7. Otherwise, find the rule which rewrites the new X_n^s. Replaced the starred X_n^s* in the current D rule with the categories on the right of the IRL rule. Repeat step 6.

Step 7. Test to see if any unmarked rules remain in the IRL. If they do, return to step 5. If not,

a. Erase starred categories, leaving only the star.
b. Add rules of the form X(*) for any non-starred terminal category of the IRL.
c. Add as axiom(s) of the DG the substitute(s) of the axiom(s) of the SPG.
d. Add the assignment rules of the SPG. If a category is subscripted in the DG, the assignments are duplicated for each subscripted variant.
e. Erase exponents.

At the end of step 7, we obtain the derived DG2.

Axiom: $* \, (N_1)$

Dependency Rules:

1. $N_1 \; (D \; * \, N_2)$

---

2. Then the IRL is: $A^1 \rightarrow \ldots A_*^0 \ldots$

$B^2 \rightarrow \ldots B_*^1 \ldots$

$C^2 \rightarrow \ldots C_*^1 \ldots$

$B^1 \rightarrow \ldots B_*^0 \ldots$

$C^1 \rightarrow \ldots C_*^0 \ldots$

2. N_2 (V D * [N])

3. N (D *)

4. V (*)

5. D (*)

If we interpret each distinct X_n as a separate category, then the same list of words is assigned to the three categories, N, N_1, and N_2, by the assignment rules. If we interpret them as the same category, then the subscripts distinguish the different substructures of a sentence in which the N's occur. Each N governs a D directly on the left, but if it is the axiom it is required to govern another N on the right also. If N is not the axiom but is governed directly by the axiom, it <u>may</u> govern another N on the right, and is required to govern a V on the left. If it is neither the axiom nor a direct dependent of the axiom, it governs nothing but the D. We may not erase the subscripts and write a single conflated rule N ([V] D * [N]), for then strings not generated by SPG2 would be generated. (For example, the rule would generate an infinite set of strings, $(D\ N)^n$.) In such circumstances we say that the DG is <u>structure sensitive</u>.

Definition 1. A DG is structure sensitive if
- a. the set of terminals assigned to one category is identical to the set assigned to any other category, and/or
- b. any rule restricts the choice of dependents a category may govern to a subset of the ordered dependents it is permitted to govern in some other rule.

Note that a DG containing the rules A (*) and A (B * C) is structure sensitive by this definition. Here, too, conflation is impossible, since A ([B] * [C]) allows A to govern B without governing C.

This structure-sensitive feature of some DG apparently serves functions like those served by the context-sensitive feature of context-sensitive phrase structure grammars in placing restrictions on the string generated, but there seems to be no mention of it in the literature. Its character may be masked by the freedom to set up categories and assign the same terminals to them.[1] For example, one may substitute simple symbols X and Y for the complex symbols N_1 and N_2, and obtain the rules

X (D * Y)
Y (V D * [N])
N (D *)

[1] Gaifman's method [4] for converting SPG to DG makes great use of this freedom.

In this case, only the assignment rules, assigning exactly the same set of terminals to X, Y, and N, explicitly show the structure sensitivity, although the fact that N governs a subset of the dependents of X and Y is significant.

Such arbitrariness in assigning symbols raises the significant linguistic problem of criteria for establishing categories, which is too large an issue to be discussed here. The problem is no less relevant to SPG.

Definition 2. An SPG is structure sensitive if

a. the set of terminals assigned to one terminal category is identical to the set assigned to any other, and/or
b. any rewriting rule does not contain, on the right, a unique category (other than the axiom) which occurs only once in that rule and appears on the right in no other rule.

The linguistic implications of the property may be clarified by considering two sets of rules, one for the artificial language a^n b a^n and one for a fragment of English.

The language a^n b a^n is generated by the rules S $\rightarrow$ A S A, S $\rightarrow$ B, A $\rightarrow$ a, and B $\rightarrow$ b. The first rule "does not contain a unique category (other than the axiom) which occurs only once", since A occurs twice. One of the A's must be starred in converting to a DG and the DG will distinguish two categories of a, a left A and a right A. Note that the same language is generated if the first rule is S $\rightarrow$ A S and a transformational rule X B => X B X is added. In this case, each rewriting rule contains a unique category other than the axiom, and a structure-free set of dependency rules is obtainable from them.

A less artificial but still extreme case of structure sensitivity is that in which two or more rules are rewritten in exactly the same way. Assume that an SPG for English has the following rewriting rules:

S $\rightarrow$ X VP
VP $\rightarrow$ V Y
X $\rightarrow$ D N
Y $\rightarrow$ D N

Here the distinction drawn between a sequence D N that <u>is</u> <u>a(n)</u> X (i. e., derives from X) and a D N that <u>is</u> <u>a</u> Y, reflects the functional notions of subject and object, but obscures the categorial notion <u>is</u> <u>a</u> noun phrase. Chomsky [3, pp. 68-72] argues that it is confusing and redundant to assign categorial status to both notions since the purely relational character of the functional notion is implicit in the rewriting rules S $\rightarrow$ NP VP and VP $\rightarrow$ V [NP]; that is, the notion "subject of a sentence" refers to the NP under the immediate domination of S, and "object of a verb" refers to the NP under the immediate domination of VP. Chomsky also shows that for sentences like "John was persuaded by Bill to leave" where "John" is simultaneously _object-of_ "persuade" and _subject-of_ a transformed embedded sentence "John leave", it is impossible to represent such functional notions by categorial assignments, and adds that "Examples of this sort, of course, provide the primary motivation and empirical justification for the theory of transformational grammar." [p. 70].

Whether it is possible or desirable to require that SPG components of transformational grammars for natural languages be structure free is an open question. Presumably, it is desirable if it is possible, since the least powerful, most restricted grammar -- the tightest fit -- is to be preferred. Moreover, inspection of proposed grammars, for English at least, indicates that most of their rules do contain unique categories on the right.

Returning now to SPG2, we see that it is structure free, since every elementary rewriting rule for every category X_i contains a single occurrence of at least one category Y_i on the right which does not appear elsewhere on the right, and is not the axiom. Under these conditions we will say that Y_i is a _head_ for category X_i and call all such Y's _head categories_. Examination shows that the structure-sensitive property of DG2 arose from the choices made in augmenting SPG2. If only head categories are marked, a structure-free DG similar to DG1 results, in which all signs of augmentation can be erased without altering its generative capacity.

Intuitively, it seems reasonable to regard heads as sources of a "governor" in any string derivable from the category in whose rewriting rules they appear. This does not mean that the string is to be considered endocentric in the strong sense of requiring that the governor be substitutable for the entire string without loss of grammaticality, and the objection sometimes raised that dependency theory forces a purely endocentric analysis of a language is based on failure to distinguish between "head of" and "substitutable for". It appears truer to say that dependency analysis assumes that one phrase type is distinguished from another primarily by the singular presence of some category in it rather than by co-occurrence and order of categories in it.

Incidentally, aside from the problem of obtaining a structure-free DG from an SPG, avoidance of structure sensitivity may be a criterion for assigning government when one is analyzing a language in terms of dependency theory. In English, for example, the choice has generally wavered between noun and verb as candidates for sentence government. Since every elementary sentence contains one verb but may contain several nouns, choosing the noun forces a structure-sensitive DG.

In converting a DG to an SPG, no requirement of intrinsic ordering needs to be imposed on the dependency rules, as it was on the rewriting rules of SPG. Dependency rules may always be partially ordered by starting with the rule (or rules) for the axiom(s). Call these "level zero rules". Then level 1 rules are those which assign dependents to the axiom, level 2 those which assign dependents to categories which make their first appearance in level 1 rules, and so on. To insure eventual termination, however, it is required that if X occurs anywhere in a level n rule, and is a dependent in a level m rule, $m \geq n$, then its choice as dependent in the level m rule is optional or else the governor in the level m rule is optionally chosen at some point. Otherwise no constraints on recursion are necessary, and any category may be reintroduced at a lower level.

By contrast, the problem of conflation arises. It has been shown that rules like A (*) and A (B * C), occurring in structure-sensitive DG, cannot be conflated. Conflations of A (B *) with A (C * D) and of A (* B) with A (B *) are also impossible, although structure sensitivity as defined above is not involved. If the rules are not conflatable because the DG is structure sensitive, then the SPG may also be structure sensitive. If they are not conflatable solely because of disparate number or position (left or right) with respect to the governor, the SPG will be structure free. In either case, the conversion process becomes somewhat more complicated.

The process will be illustrated first by applying it to a DG whose rules cannot be fully conflated both because some are structure sensitive and because some assign disparate numbers or positions to dependents of a category.

Method 2. Conversion of DG to SPG

Step 1. Augmentation

Definition: A <u>dependent element</u> in a dependency rule is any braced or bracketed string not included in larger braces or brackets, and any single unbraced, unbracketed occurrence other than * occurring within parentheses.

E.g., in A $\left(\begin{Bmatrix} B\ [C] \\ D \end{Bmatrix} * E\ [F]\right)$ there are three dependent elements: $\begin{Bmatrix} B\ [C] \\ D \end{Bmatrix}$, E, and [F].

a. Precede each dependent element with a coefficient $n \geq 1$, so that for any $n > 1$, at least one element is preceded by $n - 1$, and for any $m > n$, m does not intervene between n and *. Thus:

A (2B * 1C)
A (2B 1C *)
but not A (2B * 2C)
and not A (1B 2C *).

b. Assign an exponent s to the governor in each rule, where s equals the largest coefficient n of any dependent. (For rules of the form X (*), s is zero.)

c. If a category X is assigned different exponents for different rules in which it occurs as governor, associate a distinct subscript with each distinct exponent.

d. Replace every occurrence of X as dependent by X^s. If X has been subscripted, include each variant in braces, thus:

$$\begin{Bmatrix} X_1^{s_i} \\ X_2^{s_j} \end{Bmatrix}$$

Step 2. Test for unmarked (unprocessed) rules. If none remain, go to step 4. Otherwise take the first unmarked rule, where some $X_n^{s_i}$ is governor and set a counter $S = s_i$.

Step 3. If S = 0, mark the rule and repeat step 2. If $S \neq 0$, construct a rule of an Intermediate Rule List of the form $X_n^S \rightarrow \ldots$. On the right of the arrow duplicate all dependent elements whose coefficients n = S. Include the *, and precede it with X_n^{S-1}. Decrease S by 1, and repeat step 3.

Step 4. Establish a Table of Substitutes, assigning a unique symbol to every distinct $X_n^{S_i}$ for which $S_i \neq 0$.

Step 5. Assign the substitute or substitutes for the axioms of the DG as axioms of the SPG.

Step 6. Rewrite the IRL, using substitutes from the TS, deleting exponents and subscripts from any X_n^0, and omitting *.

Step 7. Transfer the assignment rules.

This method will be illustrated using an abstract DG3, without assignment rules. DG3 is structure sensitive since there are rules which restrict the choices of dependents of categories A and C to subsets of the ordered categories they are permitted to govern in other rules.

DG3 (augmented)

Axiom: * (A)

Dependency Rules:[1]

1. A (1B *)
2. $A\ (2 \begin{Bmatrix} B\ \ D \\ C \end{Bmatrix}\ 1B * 3E)$
3. B (1E *)
4. B (* 1F)
5. C (1G * 1F)
6. C (*)
7. D (*)
8. E (* 1G)
9. F (1A *)
10. G (*)

The final augmented form after step 1 is:

$$\text{Axiom:}\quad * \begin{Bmatrix} A_1^1 \\ A_2^3 \end{Bmatrix}$$

Dependency Rules:

1. $A_1^1\ (1B^1\ *)$
2. $A_2^3\ (2 \begin{Bmatrix} B^1\ \ D^0 \\ C_1^1 \\ C_2^0 \end{Bmatrix}\ 1B^1 * 3E^1)$
3. $B^1\ (1E^1\ *)$
4. $B^1\ (*\ 1F^1)$
5. $C_1^1\ (1G^0 * 1F^1)$
6. $C_2^0\ (*)$
7. $D^0\ (*)$
8. $E^1\ (*\ 1G^0)$

---

[1]Note the impossibility of further conflation, and that every category occurring as dependent in a rule of level m and also occurring in some other rule of level $n \leq m$ is always optionally chosen as a dependent. That is, there is the possibility of avoiding its reintroduction. Thus A, the axiom, is reintroduced in rule 9, but its governor, F, is optionally chosen as a dependent. Otherwise no derivation could terminate.

9. $F^1 \; (1 \left\{ \begin{matrix} A_1^1 \\ A_2^3 \end{matrix} \right\} *)$

10. $G^0 \; (*)$

At the end of iterations on steps 2 and 3, the IRL is

1. $A_1^1 \rightarrow B^1 \; A_1^0 *$
2. $A_2^3 \rightarrow A_2^2 * \; E^1$
3. $A_2^2 \rightarrow \left\{ \begin{matrix} B^1 \; D^0 \\ C_1^1 \\ C_2^0 \end{matrix} \right\} A_2^1 *$
4. $A_2^1 \rightarrow B^1 \; A_2^0 *$
5. $B^1 \rightarrow E^1 \; B_*^0$
6. $B^1 \rightarrow B_*^0 \; F^1$
7. $C_1^1 \rightarrow G^0 \; C_1^0 * \; F^1$
8. $E^1 \rightarrow E_*^0 \; G^0$
9. $F^1 \rightarrow \left\{ \begin{matrix} A_1^1 \\ A_2^3 \end{matrix} \right\} F_*^0$

A possible TS is:

$S : A_1^1$

$Z : A_2^3$

$Y : A_2^2$

$X : A_2^1$

$W : B^1$

$V : C_1^1$

$U : E^1$

$T : F^1$

SPG3 is:

Axiom(s): $\left\{ \begin{matrix} S \\ Z \end{matrix} \right\}$

Rewriting Rules:

1. $S \rightarrow W \ A$
2. $Z \rightarrow Y \ U$
3. $Y \rightarrow \left\{ \begin{matrix} W \ D \\ V \\ C \end{matrix} \right\} X$
4. $X \rightarrow W \ A$
5. $W \rightarrow U \ B$
6. $W \rightarrow B \ T$
7. $V \rightarrow G \ C \ T$
8. $U \rightarrow E \ G$
9. $T \rightarrow \left\{ \begin{matrix} S \\ Z \end{matrix} \right\} \ F$

SPG3 is structure sensitive, since two categories, S and X, are rewritten the same way. This results from the fact that, in DG3, A has two sets of dependents and one set is included in the other. The structure-sensitive rules for C in DG3 produced no additional structure-sensitive rules in SPG3, however, since one of them, C (*), was not processed in step 3 and did not form part of the IRL.

SPG3 may be rewritten as a structure-free grammar in a purely ad hoc way by eliminating the rewriting rule for X and substituting S for the occurrence of X on the right. More generally, it is reasonable to require of the original DG that its rules be designed so that it is possible to write a single rule (schema) assigning dependents to any given category. This requirement is reasonable if the DG is a base component of a transformational grammar whose transformations take care of the eventual order of elements in a sentence. The primary function of the DG's dependency rules is, in this case, only that of listing co-occurring categories in the dependency relations in some canonical order. In DG3, A always occurs with B as dependent. Whenever E is a dependent of A, then either a second B or a C are also dependents, and if a second B is a dependent of A, D is also. This set of conditions is summed up by

$$A\left(B \ * \ \left[\left\{ \begin{matrix} B \ D \\ C \end{matrix} \right\} \ E\right]\right).$$

Similarly, B always occurs with E or F as dependents, i.e.,

$$B \left(* \left\{ \begin{matrix} E \\ F \end{matrix} \right\} \right);$$

and C occurs with no dependents or else with both G and F, i.e.,

$$C\ (*\ [G\ \ F]).$$

These rules express co-occurrence relationships more directly than the original rules do. Let us assume this constraint and re-design DG3 as DG4. Let the augmented DG4 be:

Axiom: $*\ (A)$

Dependency Rules:

1. $A \left(2B\ \ *\ \ 1 \left[\left\{ \begin{matrix} B & D \\ & C \end{matrix} \right\}\ \ E \right] \right)$
2. $B \left(*\ \ 1 \left\{ \begin{matrix} E \\ F \end{matrix} \right\} \right)$
3. $C\ (*\ \ 1[G\ \ F])$
4. $D\ (*)$
5. $E\ (*\ \ 1G)$
6. $F\ (1A\ \ *)$
7. $G\ (*)$

The IRL is:

$$A^2 \rightarrow B^1\ \ A^1_*$$

$$A^1 \rightarrow A^0_* \left[\left\{ \begin{matrix} B^1 & D^0 \\ & C^1 \end{matrix} \right\}\ \ E^1 \right]$$

$$B^1 \rightarrow B^0_* \left\{ \begin{matrix} E^1 \\ F^1 \end{matrix} \right\}$$

$$C^1 \rightarrow C^0_*\ \ [G^0\ \ F^1]$$

$$E^1 \rightarrow E^0_*\ \ G^0$$

$$F^1 \rightarrow A^2\ \ F^0_*$$

With appropriate substitutions, this becomes SPG4:

Axiom: Z

Rewriting Rules:

1. $Z \rightarrow X\ \ Y$

2. $Y \rightarrow A \left[\begin{Bmatrix} X\ D \\ W \end{Bmatrix} \ V \right]$

3. $X \rightarrow B \begin{Bmatrix} V \\ U \end{Bmatrix}$

4. $W \rightarrow C\ [G\ \ U]$

5. $V \rightarrow E\ \ G$

6. $U \rightarrow Z\ \ F$

SPG4 is structure free. Furthermore, it has only one axiom, whereas SPG3 had two even though derived from a DG which had only one.

The additional restrictions proposed for DG and SPG in discussions of the results of conversion by these methods are not crucial as far as obtaining systematically corresponding grammars is concerned. Without them, every complete subtree of a D-tree will correspond to a complete subtree of a P-tree over the same string, and every complete subtree of the P-tree will correspond to a connected subtree of a D-tree over the same string. (No formal proof of this was given, but the methods of constructing an IRL make it moderately apparent.) Hays [5] suggests the term <u>relational correspondence</u> for this state of affairs. Also there is a systematic relationship between the categories of the two grammars, which the TS makes explicit.

Sometimes that relationship is simple, as in the case of DG4 and SPG4. Given any category of DG4, there is exactly one category of SPG4 from which the same set of strings is derivable. The relationship also holds between the categories of DG1 and SPG1, and between those of DG2 and SPG2. Under these conditions, Hays [5] calls the categories "substantively equivalent". The relationship between the categories of DG3 and SPG3 is less simple. There the set of strings derivable from A of the DG is the union of the set of strings derivable from S and Z of the SPG, and the set derivable from B is the union of the set derivable from W and V.

Hays says that a D-tree and a P-tree <u>correspond</u> if they correspond relationally and if the category at the origin of every complete subtree of the D-tree is substantively equivalent to the category labeling the complete subtree of the P-tree related to it. If a DG and an SPG "have the same terminal alphabet, and for every string over that alphabet, every structure attributed by either corresponds to a structure attributed by the other", he calls the two grammars "strongly equivalent". [5, p. 521] We prefer to say that they <u>correspond substantively</u>, since relational correspondence is asymmetric and there are always "left-over" SPG categories to which no DG category is substantively equivalent. The weaker relationship exhibited by SPG3 and DG3, where some DG categories are not substantively equivalent to any single SPG category, we have been calling _systematic correspondence_.

Conversion by the methods given here results in systematic correspondence. If the suggested constraints, which appear to be linguistically well-motivated for base components of transformational grammars, are imposed on the form of the source grammar, the target grammar corresponds substantively as well as systematically to the source grammar and both are structure free.

BIBLIOGRAPHY

1. Bar-Hillel, Y., Perles, M., and Shamir, E. "On formal properties of phrase structure grammars," in R. D. Luce, R. Bush, and E. Galanter (eds.), _Readings in Mathematical Psychology_, Vol. II, pp. 75-104. New York, Wiley, 1965.

2. Chomsky, Noam. "On certain formal properties of grammars," in R. D. Luce, R. Bush, and E. Galanter (eds.), _Handbook of Mathematical Psychology_, Vol. II, pp. 323-418. New York, Wiley, 1963.

3. Chomsky, Noam. _Aspects of the theory of syntax_. The M.I.T. Press, Massachusetts Institute of Technology, Cambridge, Mass., 1965.

4. Gaifman, Haim. _Dependency systems and phrase structure systems_. P-2315, The RAND Corporation, Santa Monica, California, May 1961.

5. Hays, David G. "Dependency theory: a formalism and some observations," _Language_, Vol. 40, (Oct.-Dec. 1964), pp. 511-525.

6. Hays, David G. _An annotated bibliography of publications on dependency theory_. RM-4479-PR, The RAND Corporation, Santa Monica, California, March 1965.

7. Kay, Martin. _The tabular parser: A parsing program for phrase structure and dependency_. RM-4933-PR. The RAND Corporation, Santa Monica, California, July 1966.

8. Lees, R. B. _The grammar of English nominalizations_, Supplement to _International Journal of American Linguistics_, 26, No. 3, Part II, 1960.

9. Robinson, Jane. "A dependency-based transformational grammar." IBM Corporation, Yorktown Heights, New York. (Forthcoming, 1967)

10. Rosenbaum, Peter S. "English Grammar II." IBM Corporation, Yorktown Heights, New York. (Forthcoming, 1967)

11. Zwicky, A. M., Hall, B. C., Fraser, J. B., Geis, M. L., Mintz, J. W., Isard, S., and Peters, P. S. "English preprocessor manual." Informations System Language Studies Number Seven, SR-132, The MITRE Corporation, December 1964.