Preprint No. 47

Classification: AL 3.3

PROPERTIES OF FORMAL GRAMMARS WITH MIXED TYPE OF RULES AND THEIR LINGUISTIC RELEVANCE

- A. K. Joshi -

INTERNATIONAL CONFERENCE

ON

COMPUTATIONAL LINGUISTICS

⇓

COLING

1969

KVAL
Forskningsgruppen
för
KVANTITATIV
LINGVISTIK

RESEARCH GROUP FOR QUANTITATIVE LINGUISTICS

Address: Fack Stockholm 40, SWEDEN

PROPERTIES OF FORMAL GRAMMARS WITH MIXED TYPES OF RULES AND THEIR LINGUISTIC RELEVANCE

Aravind K. Joshi*

ABSTRACT

In this paper, we will study a class of formal grammars with mixed types of rules. The reason for considering such grammars is that no single style (i.e., formal character of rules) of formal grammars is able to represent the various aspects of language structure in a natural way. Various considerations for setting up such grammars have been discussed. Generation schemes which map strings in the language of one mixed grammar into strings in the language of another mixed grammar (both strings being 'well-formed') have been studied. Linguistic relevance of these concepts has also been discussed.

---

* University of Pennsylvania, Philadelphia, Pa. U.S.A.

1. Introduction

A study of formal properties of different styles of formal grammars is of great interest because each style (i.e., formal character of rules) is well suited for characterizing certain aspects of natural language structure and is awkward for characterizing certain other aspects. The awkwardness can be due to either an inherent difficulty in characterizing a certain aspect (e.g., the characterization of the notion of the 'head' of a constituent in a PSG) or an unnecessary complexity in characterizing a certain aspect (e.g., the statements concerning the relational aspects in a PSG) or actually a counterintuitive characterization (e.g., this often happens in a PSG, especially in the context of transformational grammars, because a PSG allows an 'uncontrolled' introduction of new 'nonterminals'). This naturally suggests a study of formal grammars of mixed types in order to take advantage of different styles.* Thus we try to see how far we can succeed in setting up a class of grammars which has no more power than necessary and which also can characterize different aspects of natural language structure in a natural way.

The class of grammars studied here and in Joshi (1969) have been motivated by the type of grammar proposed by Harris (1962, 1968). These grammars also arose out of an attempt to formalize certain aspects of the type of grammar considered by Joshi (1966) where it was used for defining structures for the purpose of defining transformations and ultimately for constructing a transformational decomposition procedure.

First, in Section 2, we will introduce a new style of formal grammars called String Adjunct Grammars (AG). The only purpose of Section 2 is to state some of the basic concepts and results concerning AG's (including a brief discussion of their linguistic relevance) which are needed for the presentation of the material in Section 3 (for a detailed treatment of AG's, see Joshi, Kosaraju, Yamada (1968)). In Section 3, we will introduce a class of grammars called Mixed String Adjunct Grammars (MAG) which use two different types of rules - adjunction rules and a special type of rewrite rules. After studying some properties of MAG's we introduce Generation Schemes $G_s = (G, \Delta)$. A G_s maps strings in the language, L(G), corresponding to an MAG, G, into strings in the language, L(G'), corresponding to another MAG, G'.

---

* See also Robinson (1968) for a similarly motivated work.

Strings in L(G) and L(G') are both 'well-formed'. In Section 3.5 we discuss briefly the linguistic relevance of the material in the earlier sections. A detailed development of the various ideas introduced here will be reported in Joshi (1969).

Fig. A. at the end summarizes the hierarchy of some subclasses of AL's and MAL's in relation to the phrase structure hierarchy.

2. String Adjunct Grammars (AG)

Briefly an AG consists of a finite alphabet, a finite set of strings on this alphabet and a finite set of adjunction rules which state how certain adjunct strings are adjoined to certain host strings. The corresponding language called a String Adjunct Language (AL) is then defined as the set of all strings derived from a certain specified subset of the given set of finite strings. The rules in an AG have a considerably different formal character as compared to the 'rewrite rule' in a general phrase structure grammar (PSG). The language hierarchies of AG's and PSG's cut across in many interesting ways.

2.1 Local String Adjunct Grammar

We will define a local string adjunct grammar (LAG) as follows. Let $A = \{a_1, a_2, \ldots, a_m\}$ be a finite alphabet. Let Σ be a finite set of finite strings on A and let $\Sigma_c \subseteq \Sigma$ be a distinguished set of strings on A. We will call Σ the set of basic strings and Σ_c the set of basic center strings. We will define a local left adjunction rule, ℓ_{ijk} as a 3-tuple $(\sigma_i, \sigma_j, \ell_k)$ where $\sigma_i \in \Sigma$, $\sigma_j \in \Sigma$ and ℓ_k is a point of adjunction in σ_i. We will call σ_i as the (basic) host of ℓ_{ijk} and σ_j as the (basic) adjunct of ℓ_{ijk}. The point of adjunction ℓ_k of ℓ_{ijk} refers to the point of adjunction which is to the left of the kth symbol of the host σ_i where we associate with each string $\sigma_i \in \Sigma$, $\sigma_i = a_{i_1} a_{i_2} \cdots a_{n_i}$; $a_{i_j} \in A$, and $j = 1, 2, \ldots, n_i$, $2n_i$ points of adjunction, one to the left and one to the right of each a_{i_j}. Note that $\sigma_i \neq \varepsilon$, the null string. A local right adjunction rule r_{ijk} is similarly defined as a 3-tuple $(\sigma_i, \sigma_j, r_k)$, $\sigma_i \in \Sigma$, $\sigma_j \in \Sigma$ and r_k is the point of adjunction of r_{ijk} and refers to the point of adjunction to the right of the kth symbol in the host σ_i. In general, $(\sigma_i, \sigma_j, \xi_k)$ will denote a local adjunction rule, u_{ijk}. If u_{ijk} is a local left adjunction rule then $\xi_k = \ell_k$ and if u_{ijk} is a local right adjunction rule then $\xi_k = r_k$. Finally, we have the following

Definition 2.1.1 A local string adjunct grammar (LAG), G is a 6-tuple, $G = (A, \Sigma, \Sigma_c, \Sigma_h, \Sigma_a, J)$ where A is the alphabet, Σ is the set of basic strings, Σ_c is the set of basic center strings, Σ_h is the set of basic host strings, Σ_a is the set of basic adjunct strings, and J is a finite set of local adjunction rules. $\Sigma_h = \{\sigma_i | (\sigma_i, \sigma_j, \xi_k) \in J\}$, $\Sigma_a = \{\sigma_j | (\sigma_i, \sigma_j, \xi_k) \in J\}$, and $\Sigma = \Sigma_c \cup \Sigma_h \cup \Sigma_a$. Further Σ_c may contain ε but $\varepsilon \notin \Sigma_h$.

Given J, Σ_h and Σ_a are completely specified and $\Sigma = \Sigma_c \cup \Sigma_h \cup \Sigma_a$. Further the alphabet need not be explicitly stated. Hence, unless otherwise necessary we will write G as a pair (Σ_c, J) instead of a 6-tuple as in the definition above.

Example 2.1.1 Let $G = (\Sigma_c, J)$ where $\Sigma_c = \{abc\}$, and $J = \{u_1 = (abc, pq, r_1), u_2 = (pq, pq, \ell_2)\}$. [We will write u_{ijk} as just u. The indexing of u's in J is arbitrary and is merely for convenience.] Here $\Sigma_c = \{abc\}$, $\Sigma_h = \{abc, pq\}$, $\Sigma_a = \{pq\}$, $\Sigma = \{abc, pq\}$. Note that abc is a basic center string but pq is not. u_1 is a local right adjunction rule and u_2 is a local left adjunction rule. Here $A = \{a, b, c, p, q\}$.

2.2 Local String Adjunct Language (LAL)

The meaning of an adjunction rule, say, $u = (\sigma_i, \sigma_j, \ell_k)$ is that from σ_i we can derive a new string by adjoining σ_j to the left of the kth symbol in σ_i. Thus, for example if $u = (abc, t, \ell_2)$ we can derive a string atbc. However, in order to define the language L(G) corresponding to a given LAG, G, we must first define how the rules of adjunction are extended to derived (i.e. non-basic) host strings and adjunct strings. Here we will give an example to illustrate the main idea and omit the precise definition (see Joshi, Kosaraju, Yamada (1968)).

Example 2.2.1 Consider the LAG, G, in Example 2.1.1. $\Sigma = \{abc, pq\}$, $\Sigma_c = \{abc\}$, $J = \{u_1 = (abc, pq, r_1), u_2 = (pq, pq, \ell_2)\}$. From abc by one application of u_1 we obtain apqbc. We regard the points of adjunction of apqbc to be the same as abc, i.e., the positions to the left and right of the symbols a, b and c. This apqbc is a derived host and we can apply u_1 again, obtaining apqpqbc where the newly adjoined pq is immediately to the right of a.

Again, starting with pq, by one application of u_2 we obtain ppqq. Since pq is both a basic host and a basic adjunct (in the same rule, in this example), ppqq is a derived host as well as a derived adjunct and hence it can be used as a host or as an adjunct or both in the rule u_2. This allows us to derive strings pppqqq, ppqpqq, ppqppqqq, etc. Since all of these are derived from pq they can be used as adjuncts in u_1, allowing us to derive apppqqqbc, appqpqqbc, appqppqqqbc, etc. If we use apqbc as a host in u_1, we can also derive apppqqqpqbc, appqpqqpqbc, etc. Thus we can derive, for example, from the string abc $\in \Sigma_c$ the strings apppqqqbc, appqpqqbc, apppqqqpqbc, apqpqbc, apqpqpppqqqbc, etc. All these strings will be included in the language L(G) corresponding to G.

Example 2.2.2 Let $G = (\Sigma_c, J)$, $\Sigma_c = \{ab\}$, $J = \{u_1 = (ab, ab, r_1)\}$. This grammar generates the language $L(G) = \{ w / w$ is a string on A; "the number of a's in w" = "the number of b's in w" and for any

initial proper substring of w, the number of a's is greater than the number of b's}. This language is context-free and is known to be non-linear (Schützenberger (1961)).

Remarks 2.2.1

1. In the generation of a string in L(G) we observe that once a string is adjoined to a host then the adjunct string cannot receive any further adjuncts. In other words a string which is to become an adjunct string must acquire all its adjuncts prior to its being used as an adjunct string.

2. Let w be a string in L(G) derived from some string $\sigma_i \in \Sigma_c$. The generation of w does not begin, however, with the basic center string unless, of course, w is just a basic center string itself or a center string with adjuncts which themselves do not receive any other adjuncts. We have to start from the "innermost" adjunct (adjuncts) and work our way "inside out" and finally use the basic string which is to become the center string of w.

3. During the generation if a host string receives two (or more) adjuncts then we have the two following situations. If the two adjuncts are adjoined at distinct points of adjunction of the host, then clearly those adjuncts can be adjoined in any order. However, if the two adjuncts are adjoined at the same point of adjunction of the host the order is significant. Let $u_1 = (\sigma_i, \sigma_j, \xi_k)$ and $u_2 = (\sigma_i, \sigma_m, \xi_k)$ be two rules. Let $\xi_k = \ell_1$ for example. If u_1 is used before u_2 then we obtain $\sigma_j\sigma_m\sigma_i$; but if u_2 is used before u_1 then we obtain $\sigma_m\sigma_j\sigma_i$. In other words, the adjunct adjoined later in the derivation is closer to the point of adjunction in the host (to which it was adjoined) than the adjunct adjoined earlier in the derivation.

2.3 Tree representation for a derivation in LAG

Let $G = (\Sigma_c, J)$ be an LAG. Let the rules in J be arbitrarily numbered $u_1, u_2, \ldots, u_n$. The generation tree is constructed as follows. 1) If $u_\ell = (\sigma_i, \sigma_j, \xi_k)$ is used in the derivation then we represent this as in Fig. 2.3.1a. Here we have two labeled nodes σ_j and σ_i and a directed branch from σ_j to σ_i with the label $u_\ell : \xi_k$.
2) Let a host σ_i receive more than one adjunct, say, $\sigma_{j_1}, \sigma_{j_2}, \ldots, \sigma_{j_m}$ at points of adjunction $\xi_{k_1}, \xi_{k_2}, \ldots, \xi_{k_m}$, i.e., we use rules $u_{i_\ell} = (\sigma_i, \sigma_{j\ell}, \xi_{k_\ell}) \in J$, $\ell = 1, 2, \ldots, m$. We represent this as in Fig. 2.3.1b. Now, in view of Remark 2.2.1-3 we impose a right to left ordering on the points of adjunction of a host and thus in effect define an equivalence relation on the set of derivations of a string

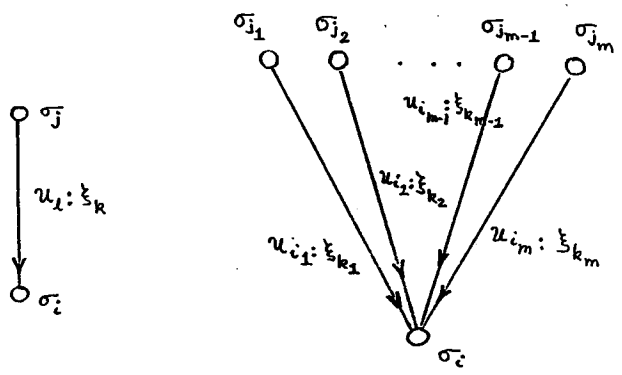


FIG. 2.3.1 a.

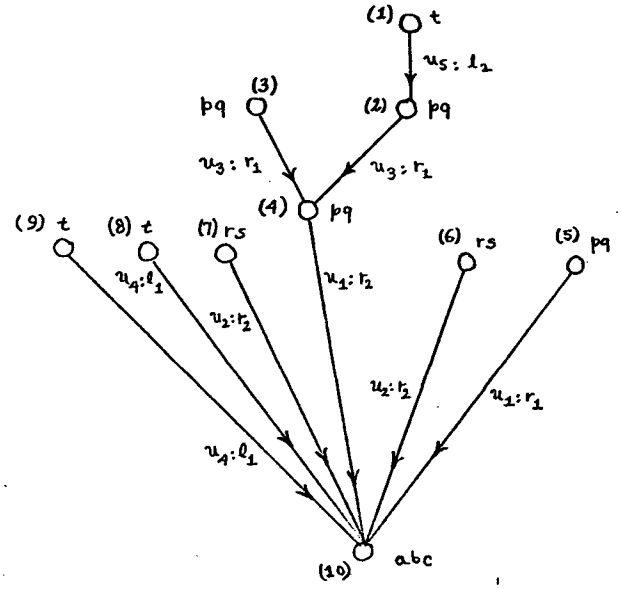


FIG. 2.3.2. r-l tree representation of a derivation of w in Example 2.3.1.

in L(G). The tree representation of a derivation with the above conventions will be called a right to left (r-ℓ) tree representation. Note that the tree representation of a derivation of a string in L(G) is a rooted tree and the string labeling the root is in Σ_c.

Example 2.3.1 Let $G = (\Sigma_c, J)$ be an LAG where $\Sigma_c = \{abc\}$, and $J = \{u_1 = (abc, pq, r_2), u_2 = (abc, rs, r_2), u_3 = (pq, pq, r_1), u_4 = (abc, t, \ell_1), u_5 = (pq, t, \ell_2)\}$. The following is a string in L(G). w = ttabrsppqptqqrspqc

Fig. 2.3.2 show an r-ℓ tree representation of a derivation of w. We have numbered the nodes for convenience. Nodes 1, 3, 5, 6, 7, 8, and 9 are terminal nodes. Node 10 is the root node.

The derived strings corresponding to the nodes of the tree in Fig. 2.3.2 are as follows: 1. t; 2. ptq; 3. pq; 4. ppqptqq; 5. pq; 6. rs; 7. rs; 8. t; 9. t; 10. w = ttabrsppqptqqrspqc.

2.4

Theorem 2.4.1 Every LAL is a CFL (context free language). The class of LAL's is properly contained in the class of CFL's. $(L = \{a^n b^n \mid n \geq 1\}$ is not an LAL.

2.5 Distributed String Adjunct Grammar (DAG) and Language (DAL)

We will generalize the local adjunction rule as follows.

Definition 2.5.1 A distributed adjunction rule, d_{ijk} is a 3-tuple, $(\sigma_i, (\sigma_j), \xi_k)$ where $\sigma_i \in \Sigma$; $\sigma_j \in \Sigma$; (σ_j) denotes a specified segmentation of σ_j; ξ_k is an adjunction 'vector', $\xi_{k_1}, \xi_{k_2}, \ldots, \xi_{k_n}$; ξ_{k_i}'s are the points of adjunction of σ_i; and $\xi_{k_i} = \ell_{k_i}$ or r_{k_i}, $1 \leq k_i \leq m$, $k_i \leq k_{i+1}$, and if $k_i = k_{i+1}$ then $\xi_{k_i} = \ell_{k_i}$ and $\xi_{k_{i+1}} = r_{k_{i+1}}$.

The meaning of d_{ijk} is that from the host σ_i we can derive a string, say, σ_p, by adjoining the segments of σ_j, i.e., $\sigma_{j_1}, \sigma_{j_2}, \ldots, \sigma_{j_n}$ at the points of adjunction $\xi_{k_1}, \xi_{k_2}, \ldots, \xi_{k_n}$ of σ_i, respectively. That is, we 'distribute' the specified segments of σ_j over σ_i at the points of adjunction $\xi_{k_1}, \xi_{k_2}, \ldots$, and ξ_{k_n}; the jth segment is adjoined at ξ_{k_j}. The condition on ξ_k prevents permutation of the segments.

The language L(G) corresponding to a given DAG, G, can be defined by generalizing the definition in Section 2.2. The main idea is that when a derived string is segmented each segment contains all the adjuncts (and adjuncts of adjuncts etc.) of all the symbols in that segment. The tree representation in Section 2.3 can also be generalized to DAG's.

Example 3.1.1 Let $G = (\Sigma_c, J)$ be a DAG where $\Sigma_c = \{abc\}$, and $J = \{u_1 = (abc, (p)\ (qr), r_1\ \ell_3), u_2 = (pqr, (p)\ (q)\ (r), \ell_1\ell_2\ell_3)\}$. Here $\Sigma = \{abc, pqr\}$, and $A = \{a, b, c, p, q, r\}$. Note that in u_1 and u_2 we have the same basic adjunct string pqr but the segmentation of pqr in u_1 and u_2 is not the same. Then

$$L(G) = \{ a\, p^{n_1+n_2+\ldots+n_m}\, b\, q^{n_1}\, r^{n_1}\, q^{n_2}\, r^{n_2} \ldots q^{n_m}\, r^{n_m}\, c \mid n_i \geq 0,$$

$$\text{for } i = 1, 2, \ldots, m;\ m \geq 1\}.$$

Example 3.1.2 Some other DAL's are: $L_1 = \{a^n b^n \mid n \geq 1\}$, $L_2 = \{a^n b^n c^n \mid n \geq 1\}$, $L_3 = \{x\, x^R \mid x \in AA^*$, x^R = reversal of $x\}$, $L_4 = \{xx \mid x \in AA^*\}$, etc.

Theorem 2.5.1* The class of LAL's is properly contained in the class of DAL's. Every DAL is a CSL (context sensitive language). The class of DAL's is properly contained in the class of CSL's. ($L = \{a^{n^2} \mid n \geq 1\}$ is a CSL but not a DAL).

Theorem 2.5.1 There are languages which are inherently ambiguous in the class of CFL's but which are unambiguous in the class of DAL's.

(Example: $L = \{a^i b^j c^k \mid i, j, k \geq 1;\ i = j$ or $j = k\}$)

2.6 String Adjunct Grammars with Null Symbols (AGN) and Language (ALN)

We will now introduce a somewhat modified form of AG's (LAG's or DAG's) called string adjuncts grammars with null symbols (AGN). The modification consists of allowing in the alphabet a very special type of "non terminal" symbols called "null symbols". The main idea is to use the null symbols to tag the strings in Σ. The null sym-

* It is possible to generalize the local adjunction rule in the following manner also. This generalization permits one to adjoin to the host a set of local adjuncts simultaneously, i.e. $(\sigma_i, \sigma_{j_1}, \sigma_{j_2}, \ldots \sigma_{j_n}, \xi_k)$ where $\sigma_i \in \Sigma$, $k = 1, 2, \ldots, n$, and ξ_k is the adjunction 'vector' as before. We will call these LAG's with simultaneous (Continued on Page 9).

bols have no points of adjunction associated with them and they do not receive any adjuncts. The null symbols are ulitmately erased (i.e., rewritten as a null string ε).

Definition 2.6.1 An LAGN (or DAGN), G, is a 7-tuple $(A, N, \Sigma, \Sigma_c, \Sigma_h, \Sigma_a, J)$ where A is a finite alphabet, N is a finite set (possibly empty) of null symbols, Σ is a finite set of basic strings, $\Sigma_c \subseteq \Sigma$ is the set of basic center strings, Σ_h is the set of basic host strings and Σ_a is the set of basic adjunct strings, $\Sigma = \Sigma_c \cup \Sigma_h \cup \Sigma_a$ and J is a finite set of adjunction rules. Further a. $A \cap N = \emptyset$; b. If $\sigma \in \Sigma$ then $\sigma \in (A \cup N)\ (A \cup N)^*$; c. There is no rule in J which adjoins adjuncts to the left or right of a null symbol, i.e., null symbols have no points of adjunction. Thus for a $\sigma_i \in \Sigma$ the adjunction 'vectors' are the same as those that can be defined for the same σ_i without the null symbols, i.e., as far as adjunctions are concerned we ignore the null symbols. We will use Greek symbols for the null symbols, and unless otherwise necessary, we will write an LAGN (or DAGN), G, as just the pair (Σ_c, J).

Theorem 2.6.1 The class of LAL's $\subseteq$ the class of LALN's and the class of DAL's $\subseteq$ the class of DALN's. (We conjecture, however, that we have "$=$" instead of "$\subseteq$").

2.7

An adjunction rule u, if applicable, can be applied arbitrarily many times. In this sense it is repeatable. We can also consider nonrepeatable rules, (nr-rule), in the sense that if a rule $u = (\sigma_i, \sigma_j, \xi_k)$ is nonrepeatable then for each occurence of the host σ_i in a derivation, u can be applied no more than once. Let nr-AG and nr-AL be the corresponding grammars (i.e. AG's which have at least one nr-rule) and languages. The class of LAL's $\subsetneq$ the class of LALN's ($L = \{a^m b^n | n \geq 1\}$ is an nr-LAL but not an LAL).

2.8

We say that a local adjunction rule is uniform if in a rule u the adjunct σ_j adjoins to the left (or right) of some symbol $a_g \in A$ in the host of u, then σ_j adjoins to the left (or right) of a_g wherever a_g occurs in any string in Σ. An LAG, G, is uniform if all its rules are uniform.

(*Continued from Page 8.)
adjunction rules (L_SAG). It can be shown that the class of LAL's $\subsetneq$ the class of L_SAL's $\subsetneq$ the class of DAL's. This observation is due to Leon Levy.

2.9

AG's with the condition $\Sigma = \Sigma_c$ are of special interest. Strings in Σ_c can be considered as elementary sentences (or sentence forms) in L(G). If $\Sigma = \Sigma_c$ then every string in L(G) can be decomposed into a set of elementary sentences (or sentence forms). Note also that if $\Sigma = \Sigma_c$ then in the r-ℓ tree representation of the derivation of string in L(G) every node is either a sentence or a derived sentence.

2.10

In an LAG (or DAG) we do not have nonterminals in the sense of the nonterminal alphabet of a PSG. We have, however, auxiliary symbols used implicitly such as the ξ_k's corresponding to the points of adjunctions. But these auxiliary symbols are used purely as position markers and do not have the same interpretation as the nonterminals in a PSG (i.e., the auxiliary symbols ξ_k's do not correspond to phrase types). If we consider the marking symbol, $\wedge$, used in making precise the definition in Section 2.2, (see Joshi, Kosaraju, Yamada (1968)), also as an auxiliary symbol then one can possibly consider $\hat{a}_i$ ($a_i \in A$) as a nonterminal which can be interpreted as a phrase type but with the added interpretation that a phrase type a_i has $\hat{a}_i$ as the 'head' (or 'center') of the phrase. Each sentence in L(G) has also a 'center' namely the center string of w.

In an LAGN (or DAGN) the null symbols are, however, like the nonterminals in the PSG although highly restricted. The null symbols are used to tag basic strings and therefore they are not used as position markers; in fact, they have no positional interpretation. The null symbols as nonterminals are highly restricted because they are never 'rewritten' (in the sense of a PSG) into any other string except the null string, i.e., the only 'rewrite rule' associated with a null symbol, say α, is $\alpha \rightarrow \varepsilon$.

LAGN (or DAGN) can be considered as grammars of a mixed style as we use rules of two different styles - adjunction rules and 'rewrite rules' of a special type. This is a very simple example of a mixed grammar. In Section 3 we will be considering some more interesting classes of mixed grammars.

2.11

In the linguistic context the alphabet A in an AG, G, will consist of symbols which denote major dictionary classes (lexical classes) such as N (nouns), t (tense, auxiliaries), A (adjectives), V (verbs), P (prepositions), wh (who, which, whom), R (pronouns), D (adverbs), Q (quantifiers), etc. N, t, A, V, etc. are thus

preterminal symbols. The basic center strings thus correspond to basic (elementary) sentence forms, e.g., N t V (John came), N t V N (Jim bought books), N t V P N (people rely on John) , etc. (A subcategorization of V's is implied here and is not explicitly shown). Basic adjunct strings are basic adjunct forms, e.g., P N (from Philadelphia), A (old), wh N t V (whom John saw), wh t V N (who saw Jim), D (quickly), etc. Each derived string in L(G) is thus a derived sentence form, e.g., (assuming suitable adjunction rules), N P N t V N (a man from Philadelphia bought books), A N t V (an old man came), N wh N t V t V D (the man whom Bill saw ran quickly), N wh N wh t V N t V t V D (the books (which) the man who met Jim bought will arrive soon) , etc. (ignoring articles for simplicity).

In an AG, lexical insertion takes place as each basic string is brought into the generation of a sentence. Let $\sigma_i = a_{i_1} a_{i_2} \ldots a_{i_m}$; $a_{i_j} \in A$ be a basic string. As σ_i is brought into the generation of a sentence, each a_{i_j} can be 'rewritten' as a set, say, α_j, of syntactic features and dictionary items can be inserted immediately. The verification of selectional restrictions that hold within the domain of a basic string can be immediately carried out as any pair of adjacent symbols of σ_i are contiguous at this stage. If the basic strings are properly chosen then most selectional restrictions are brought to bear within the domain of some basic string, and indeed it turns out that basic strings (with reasonable linguistic interpretations) can be set up in this way.

There are some restrictions which hold between a host and an adjunct string; e.g., in N wh N t V t V (the man whom John met arrived), wh N t V is an adjunct of N t V and the N in N t V is really the 'object' of V in wh N t V. Some other examples are: Zeroing in conjoined sentences, e.g., everyday, he runs and swims; he played tennis but she didn't, etc. Restrictions between successive adjuncts at the same point of adjunction of the host (ordering restrictions) as in I am looking for a book with a green cover which was lying here somewhere. Restrictions between a host and two or more adjuncts at different points of adjunction of the host as in boys who can swim distrust boys who can't. All these can be easily verified.

AG's are well suited for formulating the 'endocentric' properties in the sense that this aspect of a constituent can be explicitly brought out in the structural description. There are, however, constituents which are not 'endocentric'. These are 'exocentric' in the sense that we cannot replace them by any word of a characterizing category contained in them such that the constituents can be considered as constituent expansions of the characterizing category; e.g., who will represent us at the meeting in who will represent us

at the meeting is unclear, etc. AG's are not well suited for formulating the exocentric properties. These properties are better characterized by the use of a 'nonterminal' and 'rewrite rules' in the sense of a PSG (see Section 3).

Sentence adjuncts (e.g., in general, today) can be handled well in an AG; in particular, that these adjuncts can occupy various sentence positions can be easily characterized in an AG. This is awkward to characterize in a PSG. However, the property that these adjuncts are adjuncts of a sentence is better characterized by the use of a nonterminal.

Distributed adjunction rules are required to handle cases such as two and three are even and odd numbers respectively which is a case of an intercalated structure. Such structures are not too frequent. However, if one tries to construct AG type grammars as base for transformational grammars then the need for intercalated structures is not so marginal. This is primarily because one tries to relate each adjunct to an elementary sentence (i.e., one tries to constitute the adjunct and host strings in such a way that the underlying elementary sentence(s) could be reconstructed from them). Some examples are: the man who came ... (double underline indicates the distributed adjunct); John's proof of the theorem, etc. (see Hiż and Joshi (1967), Joshi (1966, 1969), for further details). The kinds of intercalated structures possible in a DAG apparently are adequate for this purpose.

If $\Sigma = \Sigma_c$ then each string $w \in L(G)$ has a representation in terms of basic 'sentences' (or basic center strings). In general, adjuncts are not strings in Σ_c and hence $\Sigma \neq \Sigma_c$. But what seems to be true of a natural language is that one can 'almost' construct an AG, G, (actually, a mixed AG, see Section 3) for which $\Sigma = \Sigma_c$ and define a set of operations (these consist of permutations, deletions, and additions of constants; these operations can be related to transformations in a given language) for each $\sigma \in \Sigma$ and for strings derived from σ such that we can construct a new AG, G', with basic strings Σ', Σ_c' where $\Sigma' \neq \Sigma_c'$ and strings in Σ' are transformationally derived from strings in Σ. Strings in L(G') except for morphophonemic operations are the strings (sentences) in the language. In transformational analysis we go in the reverse direction, i.e., from G' to G and reconstruct the set of basic 'sentences' (together with the derivation in G) underlying a given sentence generated by G'. (See Joshi (1969) for further results and details.)

3. Mixed Grammars

3.1

In AGN's (i.e., AG's with null symbols) in Section 2.6 we use a very special type of null symbols which are ultimately erased.* The only rewrite rule associated with a null symbol is $\alpha \rightarrow \xi$ where α is a null symbol and ξ is the null string. The AGN's are thus a class of mixed grammars as rules of more than one style are used. It is, however, a rather simple class of mixed grammars.

3.2 Mixed String Adjunct Grammar (MAG)

We will now consider a more interesting class of mixed grammars. The main idea is to allow a single 'nonterminal' (in the sense of a PSG) in an AG and a special type of 'rewrite rules' associated with this nonterminal. We will, however, call them 'replacement rules'. The reason for adopting this new terminology will become clear later. More specifically, we will define a Mixed String Adjunct Grammar (with replacement rules) (MAG), G, as follows.

First, in addition to the 'terminal' alphabet A we will have a 'nonterminal' S. The set of basic strings, Σ, and the set of basic center strings, Σ_c, will now be strings on $(A \cup \{S\})$. Thus a string $\sigma_i \in \Sigma$ may now contain one or more S's in it. A $\sigma_i \in \Sigma$ which does not contain S will be called an elementary basic string and a $\sigma_i \in \Sigma$ which contains one or more S's will be called a complex basic string. The adjunction rules (local or distributed) are defined as before and we adopt the same convention as in the case of the null symbols, i.e., in an adjunction rule if the host is a complex basic string we disregard the S symbols in it as far as points of adjunction are concerned. Thus the points of adjunction of a string, say, σ_i = aS bS are the points of adjunction which are to the left and right of a and b. Further, the S symbols have no points of adjunction. Of course, if an adjunct string, say, σ_j is a complex basic string and has a specified segmentation, then each symbol S in σ_j must be included in some segment of σ_j.

---

* Actually, it is possible to define the recursive extension of the adjunction rules such that the null symbol associated with any basic string is erased at the time the string is adjoined to some host string. The null symbol associated with the center string is then erased at the end.

We now define a replacement a rule p_{ij} (often written just as p) as a pair $< \sigma_i, \sigma_j >$ where $\sigma_i \in \Sigma$, σ_i is a complex basic string, and $\sigma_j \in \Sigma_c$. The meaning of a replacement rule $p = < \sigma_i, \sigma_j >$ is that from σ_i one can derive a new string by replacing some occurence of S in σ_i by σ_j. Thus we have the following

Definition 3.2.1 An MAG, G, is a 9-tuple, $G = (A, S, \Sigma, \Sigma_c, \Sigma_h, \Sigma_a, \Sigma_r, J, R)$ where A is the alphabet (terminal), S is a 'nonterminal' symbol ($S \notin A$), Σ is the set of basic strings, Σ_c is the set of basic center strings, Σ_h is the set of basic host strings, Σ_a is the set of basic adjunct strings, Σ_r is the set of basic replacer strings, J is a finite set of adjunction rules (local or distributed), and R is a finite set of replacement rules. $\Sigma_h = \{\sigma_i | (\sigma_i, \sigma_j, \xi_k) \in J\} \cup \{\sigma_i | < \sigma_i, \sigma_j > \in R\}$, $\Sigma_a = \{\sigma_j | (\sigma_i, \sigma_j, \xi_k) \in J\}$, and $\Sigma_r = \{\sigma_j | < \sigma_i, \sigma_j > \in R\}$. $\Sigma = \Sigma_c \cup \Sigma_h \cup \Sigma_a \cup \Sigma_r$. Σ_c may contain ε but $\varepsilon \notin \Sigma_h$.

Given J and R, Σ_h, Σ_a, and Σ_r are completely specified and $\Sigma = \Sigma_c \cup \Sigma_h \cup \Sigma_a \cup \Sigma_r$. A need not be explicitly stated. Since S is the only 'nonterminal' it need not be explicitly stated also. Hence, we will write an MAG, G, as a triple (Σ_c, J, R) instead of a 9-tuple as in the definition above.

Example 3.2.1 Let $G = (\Sigma_c, J, R)$ be an MAG where $\Sigma_c = \{abc, pq, abSc\}$, $J = \{u_1 = (abSc, (a)(b)(c), \ell_1\ell_2\ell_3), u_2 = (abc, (a)(b)(c), \ell_1\ell_2\ell_3)\}$, and $R = \{p_1 = < abSc, abSc >, p_2 = <abSc, pq >\}$.

Example 3.2.2 Let $G = (\Sigma_c, J, R)$ be an MAG where $\Sigma_c = \{aSb, c\}$, $J = \{u_1 = (aSb, (a)(Sb), r_1r_2)\}$, and $R = \{p_1 = < aSb, aSb >, p_2 = < aSb, c >\}$.

3.3 Mixed String Adjunct Language (MAL)

We now have two different styles of rules in G, namely, the adjunction rules J and the replacement rules R. If R is empty then we have an AG and we know how the generation proceeds in this case. In particular, we note the 'inside out' characteristic of the generation. If R is not empty then we have replacement rules associated with the symbol S. The generation still proceeds in an

'inside out' manner. In order to define the language corresponding to an MAG, we must state how the rules in J and R are extended to derived strings. Rather than giving a precise definition, we will illustrate the main idea by the following example.

Example 3.3.1 Consider the MAG, G, in Example 3.2.1. Starting with the complex basic string abSc and using it as a host in u_1 and the string abc as an adjunct in u_1, we obtain aabbScc. Using this as a derived host in the replacement rule p_2, we obtain aabbpqcc. But this is a string derived from abSc and therefore it can be used as a replacer string in p_1. Thus we can obtain aabbaabbpqcccc (see Fig. 3.3.1). The language $L(G) = L_1$ is

$$L_1 = \{a^n b^n c^n \mid n \geq 1\} \cup \{a^{n_1} b^{n_1} a^{n_2} b^{n_2} \ldots a^{n_k} b^{n_k} pqc^{n_k+n_{k-1}+\ldots+n_1} \mid n_i \geq 0, i=1,2,\ldots,k \quad k \geq 1\}$$

As far as rules in J are concerned we require 'inside out' generation. In order to define consistently the recursive extension of rules in J and R together it is necessary that once a replacer string replaces an S no further adjunctions or replacements can be made on it. Thus before a replacer string is used it must be completely built up (i.e., it must have received all its adjuncts and adjuncts of adjuncts etc., and all occurences of S must have received their replacements).* This was the reason for calling the rules in R as replacement rules rather than rewrite rules.

Thus the generation begins from either (a) the 'innermost' host - adjunct pair(s) or (b) the 'innermost' complex basic host - replacer pair(s) where the replacer is an elementary basic center string, or (c) both (a) and (b).

The reader may amuse himself by working out the language (L_2) corresponding to the MAG in Example 3.2.2. It is rather complicated to write it down in a parametric form.

---

* It is assumed that for every complex basic string, say, σ_i, either there is a rule $< \sigma_i, \sigma_j >$ where σ_j is an elementary basic string or there is a sequence of rules $< \sigma_{i_1}, \sigma_{j_1} >, < \sigma_{i_2}, \sigma_{j_2} >, \ldots, < \sigma_{i_n}, \sigma_{j_n} >$ where $\sigma_{i_1} = \sigma_i$, $\sigma_{j_k} = \sigma_{i_{k+1}}$, k = 1, 2, ... n-1, and σ_{j_n} is an elementary basic string. Otherwise, σ_i can be removed from G without affecting L(G).

Both L_1 and L_2 are CSL's (Context sensitive languages). They are both DAL's also. This can be shown by constructing the equivalent DAG's. Let $G_1 = (\Sigma_c, J)$ be a DAG where $\Sigma_c = \{abpqc, pq\}$, and $J = \{u_1 = (abpqc, (ab)(c), r_2r_5), u_2 = (abpqc, (a)(b)(c), \ell_1\ell_2\ell_5)\}$. Then $L(G_1) = L_1$. Similarly let $G_2 = (\Sigma_c, J)$ be a DAG where $\Sigma_c = \{acb, c\}$, and $J = \{u_1 = (acb, (a)(cb), r_1r_3), u_2 = (acb, (a)(b), r_1r_3), u_3 = (acb, (a)(b), \ell_2\ell_3), u_4 = (ab, (a)(b), r_1r_2), u_5 = (ab, (a)(cb), r_1r_2)\}$. It can be shown that $L(G_2) = L_2$. In fact, we have the following

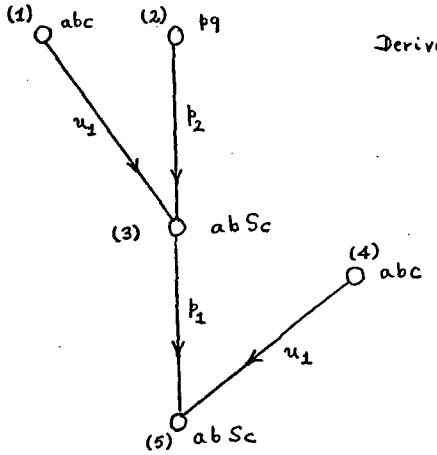
Theorem 3.3.1 For every MAG, G, there is an equivalent DAG, G', (i.e., $L(G) = L(G')$) which can be effectively found.

We will omit the proof here as it is rather long. An examination of MAG's in Examples 3.2.1 and 3.2.2 and their corresponding DAG's, G_1 and G_2 above will give the reader some indication of how the proof can be constructed. This is an interesting result because it shows that as far as weak generative power is concerned, we can eliminate S, the only 'nonterminal' we have used. It can also be shown (this is easily seen from G_1 and G_2 above) that if the MAG, G, satisfies the condition that $\Sigma = \Sigma_c$, then the equivalent DAG, G' will not necessarily satisfy the corresponding condition $\Sigma' = \Sigma_c'$. In fact, for some MAG there will be no equivalent DAG satisfying this condition (see Section 3.5.6 for linguistic relevance).

Remarks 3.3.1

1. In an MAG, G, not every basic string is a string on A (e.g., the complex basic strings). However, in the tree representation of derivation of a string in L(G), the derived strings at each node are strings on A, just as in the case of an AG. In fact, if this condition is not satisfied the tree will not correspond to a tree for some string in L(G).

2. The symbol(s) S in a complex string, say, σ_i, will be referred to as a contained S. σ_i will also be called a container string and the replacer string(s) for S will be called contained string(s). Let σ_i = abSc, σ_j = dSe, σ_k = gSh, and σ_ℓ = pq. Let u_1 = (abSc, dSe, r_1) be an adjunction rule and $p_1 = <$ abSc, gSh $>$, $p_2 = <$ abSc, pq $>$, $p_3 = <$ dSe, pq $>$, and $p_4 = <$ gSh, pq $>$ be some replacement rules. Consider the following tree representation of a derivation (Fig. 3.3.2: the superscripts on σ_ℓ are used to distinguish the two distinct occurences of the string pq). Note that σ_ℓ^1 is contained



Derived strings at the nodes 1, 2, 3, 4, and 5 are:

1. abc
2. þq
3. aabbþqcc
4. abc
5. aabbaabbþqcccc

FIG. 3.3.1. Tree representation of derivation of w = aabbaabbþqcccc in MAG, G, in Example 3.2.1.

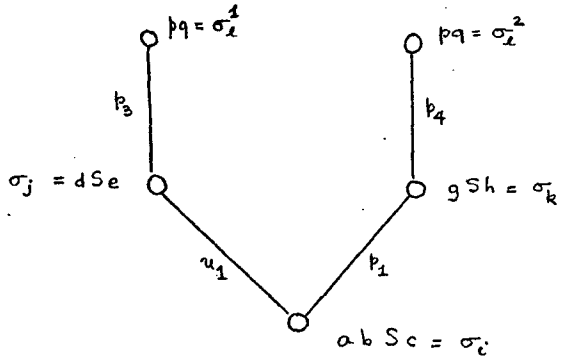


FIG. 3.3.2.

in σ_j and σ_j is adjoined to σ_i, and σ_ℓ^2 is contained in σ_k and σ_k is contained in σ_i. Both σ_ℓ^1 and σ_ℓ^2 are two 'levels' down (i.e., 'depth' 2) with respect to σ_i and this is so both in the sense of a PSG and an MAG. Now σ_ℓ^1 is two levels down where the first level is due to an adjunction and the second due to containment, but σ_ℓ^2 is two levels down where both levels are due to containment. Thus in an MAG the depth of embedding of string can be characterized not only by the number of levels involved but also by stating for each level whether it is due to adjunction or containment. There is also the possibility of characterizing an arbitrary depth of embedding in terms recurrent patterns of adjunction and/or containment levels.

3.4 Deformations and Generation Scheme

In this section we will be concerned with the construction of an MAG, G, with $\Sigma = \Sigma_c$ for an MAG, G', (for which $\Sigma' \neq \Sigma_c'$, in general) such that G is related to G' by means of certain operations (see Section 2.11 - last paragraph).

Let $\Delta = \{\delta_1, \delta_2, \ldots, \delta_n\}$ be a finite set of deformations (to be defined later).

Definition 3.4.1 A Generation Scheme, G_s is a pair (G, Δ) where G is an MAG, $G = (\Sigma_c, J, R)$, Δ is a finite set of deformations, and with each rule $u \in J$ and each rule $p \in R$, we associate unique subsets of Δ, say, Δ_u and Δ_p respectively.

Let C_o be a finite set of constants. We say that a string σ' is a deformed σ if every symbol of σ' is either a constant (i.e., is in C_o) or is a symbol of σ or both. That is, σ' is obtained from σ by one or more of the operations of permuting, deleting, or repeating symbols of σ or adding one or more symbols from C_o.

Definition 3.4.2 Let $u = (\sigma_i, \sigma_j, \xi_k) \in J$ and let Δ_u be the associated subset of deformations. Then a $\delta_i \in \Delta_u$ maps the rule $u = (\sigma_i, \sigma_j, \xi_k)$ into a 3-tuple $(\sigma_i', \sigma_j', \xi_k')$ where $\sigma_i' = \sigma_i$, σ_j' is a deformed σ_j (the specific choice of operations of permuting, deleting, or repeating symbols of σ_j or adding constants is determined by δ_i), and ξ_k' is an adjunction 'vector' of σ_i not necessarily

the same as ξ_k. We write this as $\delta_i(u) = u'$ where $u' = (\sigma_i', \sigma_j', \xi_k')$.* Similarly, a $\delta_i \in \Delta p$, $p = < \sigma_i, \sigma_j >$, maps p into a pair $< \sigma_i', \sigma_j' >$ where $\sigma_i' = \sigma_i$ and σ_j' is a deformed σ_j (δ_i determines the specific deformation as before). We write this as $\delta_i(p) = p'$, where $p' = < \sigma_i', \sigma_j' >$.

Note that δ_i deforms σ_j and also specifies a new adjunction 'vector'. Note also that u and p are rules in the MAG, G. u' and p' are not rules in G. However, they will be later interpreted as rules in another MAG, G'.

Each δ_i can then be extended to the derived hosts and derived adjuncts in u, and to derived hosts and derived replacers in p in the obvious way (i.e., if a symbol of σ_j is permuted it carries with it its adjuncts, and their adjuncts etc.; if a symbol of σ_j is deleted then we delete its adjuncts, and their adjuncts etc.; addition of constants is not affected). More complicated extensions have to defined however for the more complicated δ_i's. (See Section 3.5.5, and for further details, Joshi (1969).

For a given Generation Scheme, $G_s = (G, \Delta)$, we will define the language corresponding to G_s, $L(G_s)$, as follows. We will give here only an informal definition. We carry out the generation in the MAG, G, as described in Section 3.3, with the following modification. If during the generation we plan to use a rule u then instead of using u we use the rule $u' = \delta_\ell(u)$ where $\delta_\ell \in \Delta_u$. Similarly, if we plan to use a replacement rule p then instead of using p we use the rule $p' = \delta_m(p)$ where $\delta_m \in \Delta_p$. The choice of δ_ℓ from Δ_u and δ_m from Δ_p is nondeterministic.

---

* This definition is not quite precise as stated. First, note that ξ_k' is an adjunction 'vector' and hence its components must satisfy certain conditions (see Section 2.5). Secondly, if ξ_k' has more than one component then δ_i must also specify the appropriate segmentation of σ_j'. This definition is also weaker than required. More complicated δ_i's can be defined and are required (see Section 3.5.5).

Note that here we use the word 'language' in the usual sense, i.e., a set of strings on A. Ultimately, however, we are interested in the corresponding strings of lexical items. The lexical insertion proceeds in G in the same general manner as in Section 2.11 (paragraph 2), (see also Section 3.5). In this case the choice of a δ_ℓ from Δ_u and of δ_m from Δ_p may depend on the lexical entries for σ_i and σ_j which are, of course, available at this point in the generation.

The 'language' derived in this way is $L(G_s)$. The derivations are not in G but in G_s. We could, of course, allow the generation in G to proceed independently but concurrently with the generation in G_s. In this case, we would derive a pair of 'corresponding' strings, say w and w_s where $w \in L(G)$ and $w_s \in L(G_s)$. Note that G_s is not an MAG; however, we have the following

Theorem 3.4.1 For every Generation Scheme, $G_s = (G, \Delta)$ there is an equivalent MAG, G', (i.e. $L(G') = L(G_s)$), and G' can be effectively found.

The proof is rather involved and we will omit it. At least for the δ's in Definition 3.4.2 one can state the main idea as follows.* Let $G = (\Sigma_c, J, R)$ and $G' = (\Sigma_c', J', R')$. Then 1. $\Sigma_c' = \Sigma_c$. 2. J' obviously contains all the $u' = \delta_i(u)$, $u \in J$, as adjunction rules. But J' also contains some additional rules which are needed for the following reason. Let σ_i be an adjunct string in G and let some δ deform σ_j into σ_j'. Now, if σ_j is also a host string in some adjunction rule, say, u_r in J then we must add a new adjunction rule(s) in G' which in effect allows one to adjoin the adjunct in u_r (actually, its deformations under all possible δ's) to σ_j', with the adjunction 'vector(s)' appropriately specified. 3. R' consists of all the $p' = \delta_i(p)$, $p \in R$, as replacement rules. R' also contains some additional rules which are needed for the same reason as in 2 above.

We can impose the condition $\Sigma = \Sigma_c$ on the MAG, G. But then G' (equivalent to $G_s = (G, \Delta)$) need not satisfy a similar condition, i.e., Σ' need not be equal to Σ_c'. This is because Σ' contains, besides strings in Σ_c' ($= \Sigma_c$), the deformed adjuncts and deformed replacer strings.

---

* The proof extends to some of the more complicated δ's (see Section 3.5.5).

From Theorems 3.4.1 and 3.3.1 we then have

Corollary 3.4.1 For every $G_s = (G, \Delta)$ there is an equivalent DAG, G", (i.e., $L(G_s) = L(G")$) which can be effectively found.

3.5 Linguistic Relevance

In this section, we will briefly discuss the various results in Sections 3.1 - 3.4 in the linguistic context and provide some justifications and interpretations for these.

3.5.1

As is evident from the discussion in Section 2.11, the main motivation for considering MAG's is to provide suitable representations for certain structures (e.g., that he went home surprised me, I told him to go home, that John will come is certain, I tried to read the book, etc.). The purpose for considering MAG's with $\Sigma = \Sigma_c$ is the same as in Section 2.11 (last paragraph).

3.5.2

In Section 2.11 we have seen that many restrictions have as their domain a basic string or a basic string and its adjuncts; and these can be easily stated and, at the time of adjunction, easily verified. These remarks obviously hold for an MAG as far as adjunctions are concerned. However, in addition to these, in an MAG there are many restrictions which have as their domain a complex string and its replacer(s) string (i.e., a container string and the contained string(s)). These also can be easily stated and, at the time of replacement, easily verified. Apart from selectional restrictions, some of these restrictions are: (a) Identity of one of the N's in the container string and one of the N's in the contained string (identity of the 'subject' or 'object' noun in the container string and the 'subject' or 'object' noun of the contained string), e.g., I told John to go home, I promised Bill to purchase books, John deserves promotion, He suffered defeat, I forced him to swim, He is uninspiring as a teacher, I saw the boy being beaten by the policeman, etc. Actually, since we are considering derivations in an MAG with $\Sigma = \Sigma_c$, we should have written these somewhat as follows: I told John$^\bullet$ (John$^\bullet$ should* go home), I$^\bullet$ promised Bill (I$^\bullet$ would purchase books), John$^\bullet$ deserves (N should* promote John$^\bullet$), He$^\bullet$ suffered (N defeated him$^\bullet$), He$^\bullet$ is uninspiring (He$^\bullet$ $\not\equiv$ teach N to N), etc. ($\bullet$ marks the elements with the same reference; $\not\equiv$: untensed (or tenseless) σ). (b) Certain conditions on replacing a noun by a pronoun, e.g., John$^\bullet$ hoped (he$^\bullet$

* perhaps $\not\equiv$.

will win) but not He hoped (John will win), etc. (c) Possible correlations between tenses in the container and contained strings (see examples in (a) above).

Later, in the context of generation in $G_s = (G, \Delta)$ we will discuss some additional restrictions. These do not affect the 'well-formedness' (with respect to the satisfaction of restrictions) in G. Thus the strings of lexical items corresponding to strings in L(G) are also 'well-formed'.

3.5.3

We now consider the derivations in $G_s = (G, \Delta)$. Obviously, the purpose of δ's is to obtain the corresponding strings in G_s (i.e., also in G'), e.g., I told John to go home ← I told John (John should go home), etc. Note that for each rule u or p in G, the δ's are selected from Δ_u or Δ_p respectively. Some examples of restrictions on δ's are: (a) The choice of a δ from Δ_u (or Δ_p) may be affected by the lexical entries for the container and contained strings, mostly by the verb (including is A and is N) of the container string, e.g., I tried to drive a car, I tried driving a car, but only I stopped driving a car; That he answered the letters is true, His answering the letters is strange, but not His answering the letters is true, etc. (b) Choice of a particular preposition in a deformation may depend on the lexical entries for the container and contained (?) strings, e.g., I know of John's coming, I believe in (my) leaving early, etc. (see Section 3.5.5 and Joshi (1969) for further details).

3.5.4 A Simple Example of G, G_s and G'

Let $G = (\Sigma_c, J, R)$ be an MAG (with $\Sigma = \Sigma_c$) where A = {N, t, V, A}; $\Sigma_c = \Sigma = \{\sigma_1 = N\ t\ V, \sigma_2 = N\ t\ V\ N, \sigma_3 = N\ t\ V\ S, \sigma_4 = N\ t\ V\ N\ S, \sigma_5 = S\ t\ V\ N, \sigma_6 = S\ t\ V\ A, \sigma_7 = S\ t\ V\ S\}$*;

$J = \{(\sigma_1, \sigma_j, r_1\} \cup \{(\sigma_2, \sigma_j, r_1) \cup \{(\sigma_2, \sigma_j, r_4)\} \cup \{(\sigma_3, \sigma_j, r_1)\}$

$\cup \{(\sigma_4, \sigma_j, r_1)\} \cup \{(\sigma_4, \sigma_j, r_4\} \cup \{(\sigma_5, \sigma_j, r_3)\}$, $j = 1, 2, 3, 4$;

* A subcategorization of N's, V's, and A's is implicit here and is not shown explicitly in the notation.

and $R = \{< \sigma_i, \sigma_j > \mid i = 3, 4, 5, 6, 7;\ j = 1, 2, \ldots, 7\}$.

Let $G_s = (G, \Delta)$ be a Generation Scheme where $\Delta = \{\delta_i\}$ is a set of deformations and the δ's are defined as follows. (Set of Constants, C_o = {wh, that, to, 's, ing})

δ_1: $(\sigma_i, \sigma_j, \xi_k) \rightarrow (\sigma_i, \sigma_j^1, \xi_k)$ where $i = 1, 2, 3, 4, 5$; $j = 1, 2, 3, 4$; $(\sigma_i, \sigma_j, \xi_k) \in J$; and σ_1^1 = wh t V, σ_2^1 = wh t V N, σ_3^1 = wh t V S, σ_4^1 = wh t V N S.*

δ_2: $(\sigma_i, \sigma_j, \xi_k) \rightarrow (\sigma_i, \sigma_j^2, \xi_k)$ where $i = 1, 2, 3, 4, 5$; $j = 2, 4, 5$; $(\sigma_i, \sigma_j, \xi_k) \in J$; and σ_2^2 = wh N t V, σ_4^2 = wh N t V S, σ_5^2 = wh S t V.

δ_3: $< \sigma_i, \sigma_j > \rightarrow < \sigma_i, \sigma_j^3 >$ where $i = 3, 4, 5, 6, 7$; $j = 1, 2, \ldots, 7$; $< \sigma_i, \sigma_j > \in R$; and σ_j^3 = that σ_j.

δ_4: $< \sigma_i, \sigma_j > \rightarrow < \sigma_i, \sigma_j^4 >$ where $i = 3, 4$; $j = 1, 2, 3, 4$; $< \sigma_i, \sigma_j > \in R$; and σ_1^4 = to V, σ_2^4 = to V N, σ_3^4 = to V S, σ_4^4 = to V N S.

δ_5: $< \sigma_i, \sigma_j > \rightarrow < \sigma_i, \sigma_j^5 >$ where $i = 3, 5, 6, 7$; $j = 1, 2, 3, 4$; $< \sigma_i, \sigma_j > \in R$; σ_1^5 = (N's) V ing, σ_2^5 = (N's) Ving N, σ_3^5 = (N's) Ving S, σ_4^5 = (N's) Ving N S.

We have not shown explicitly the various subsets of Δ associated with the rules in J and R in G. But, these can be worked out from the specification of the δ's above.

---

* Actually, we should use here a distributed adjunction rule to account for the definite article to the left of N in the host. We leave this out in order not to complicate the example.

An MAG, G', equivalent to $G_s = (G, \Delta)$ can now be easily constructed (see the discussion under Theorem 3.4.1). We will not write G' here as it is too long. It is easily seen that we can derive in $G_s = (G, \Delta)$ and therefore in G' sentences such as, John wants to go home, I prefer walking, The man who came ordered Jim to shut the door, I promised Bill to tell John that he should visit Fred, Bill's forcing John to resign annoyed him, Painting the doors blue was the custom, My asking him to write a paper caused his leaving the job, etc.

3.5.5 Some Complex Deformations*

Some examples of deformations more complicated than those in Definition 3.4.2 are as follows:

a. A δ may be defined such that it requires the adjunct string, σ_j, in the rule $u = (\sigma_i, \sigma_j, \xi_k)$ to be a derived string. δ then refers to not only σ_j but string(s) which may be at most a fixed finite depth (counted in terms of adjunction and containment levels) relative to σ_j. Mostly depth 1 (and occasionally depth 2) is adequate, e.g., The man who had the keys finally came → The man finally came who had the keys: The man who finally came who had the keys ...

b. A δ may be defined as above but with the possibility of δ referring to not only σ_j but a string which is at an arbitrary depth relative to σ_j where the arbitrariness of the depth is so constrained that it can be specified in terms certain recurrent patterns of adjunction and containment levels, e.g., The meeting (which) I selected John to represent us at ..., The people we hope that John told to water the plants ..., etc. [Although slightly out of place, it might be worth mentioning here that the distinction between adjunction and containment levels also helps in stating certain pronouning restrictions to some extent, e.g., the pronoun is in the contained string and not in the container string: John• thought he• will win but not He• thought John• will win; but if we have an adjunction level then we have both: People who know Bill• like him•, and People who know him• like Bill•; if we have an adjunction level and a containment level, we again have both: People who know Bill• want to help him• and People who know him• want to help Bill•; and if we have two successive containment levels, we have John• asked Bill to tell Mary to see him• but not He• asked Bill to tell Mary to see John•, etc.]

---

* For further details see Joshi (1969).

c. A δ may be defined such that it not only deforms the adjunct string σ_j but also deforms the host string σ_i. Since the host σ_i can also be deformed by such a δ, the precise definition of how generation proceeds in the Generation Scheme, G_s, becomes complicated. Such δ's can be used to obtain from the same* container string — contained string pair, two related sentences such as, e.g., That he came surprised me and it surprised me that he came, etc.

d. Sets of related δ's to cover certain zeroings which have as their domain the container and the contained strings, e.g., I promised him to come ← I promised him that I would come, etc. One may also include here zeroing of 'appropriate' verbs, V_{app}, e.g., I expect him ← I expect him to V_{app} where V_{app} = {come, arrive, etc.}; perhaps also I shall go ← I promise you that I shall go (Harris (1968)

3.5.6

From Theorem 3.3.1 we know that for every MAG there is an equivalent DAG which can be effectively found. This means that we can eliminate the nonterminal S as far as weak generative capacity is concerned. Of course, we don't choose to eliminate S, but it is interesing to see the implications of this theorem. If one examines the proof of this theorem, we notice that in effect for every complex basic string, say, σ_i = abSc and for every elementary basic string, say, pq, which is a replacer for σ_i,** we set up an adjunction rule (in this case a distributed rule) such as (pq, (ab)(c), $\ell_1 r_2$). Thus we will have to consider I know that in I know that John went home as an adjunct of John went home. Now (in the spirit of the discussion in Section 3.4) adjuncts are obtained by deforming a string in Σ_c; also adjuncts have a certain degree of mobility within the host. This is perhaps the reason why in some cases we come close to realizing this, e.g., (1) I hope that John will win: we can obtain a sentence and a semi-sentence, John will win; I hope so or a sentence and sentence adjunct, John, I hope, will win, (2) That John passed the examination surprised me: John passed the examination; it surprised me, John passed the examination, to my surprise, etc.

---

* This avoids having two distinct strings in G generating strings which are paraphrases of each other. If, however, we allow this possibility the structure of G_s can be considerably simplified. We do not follow this approach but the nature of these simplifications is discussed in Joshi (1969). In a different context and in a different framework, Keenan (1969) has made a similar comment.

** and $\langle abSc, abSc \rangle$, and $\langle abSc, pq \rangle$ are two replacement rules,

In Section 3.4, in $G_S = (G, \Delta)$ we imposed on G the condition that $\Sigma = \Sigma_c$. Then in G' (equivalent to G_S) every adjunct string is obtained by deforming some string in $\Sigma_c (= \Sigma)$ in G. However, adjuncts such as, e.g., quite in quite forgot, very in very long, some quantifiers (all, some, etc.), some occurences of articles, some time and manner adverbials, etc. pose a problem here. There are a couple of ways around this problem.

One solution is to consider these adjuncts as primitively adjoined in G (i.e., regard them as a sort of primitive adjuncts in G)*. G, of course, will no longer quite satisfy the condition $\Sigma = \Sigma_c$.

Another more attractive solution (certainly, motivated by some current trends in transformational theory) which will maintain the condition that every basic string is also a basic center string is to construct (1) another MAG, G", by retaining all strings in Σ_c in G, excluding the primitive adjuncts in G, but adding new complex basic center strings (these will now more and more become infra-sentence forms), and also adding new adjunction and replacement rules, and (2) a new Generation Scheme $G_S" = (G", \Delta")$, where $\Delta"$ is a new deformation set, and G" satisfies the condition $\Sigma" = \Sigma_c"$, such that $G_S"$ is equivalent to G. (At this point, we may also remove the tenses, auxiliaries, and prepositions. Basic strings in G" will then be strings of N's, V's (including is A and is N) and S's). Thus we have the alternating sequence of MAG's and Generation Schemes**.

$$G": \; G_S" = (G", \Delta") \approx G: \; G_S(G, \Delta) \approx G'$$

and

$$L(G") : L(G_S") = L(G) : L(G_S) = L(G')$$

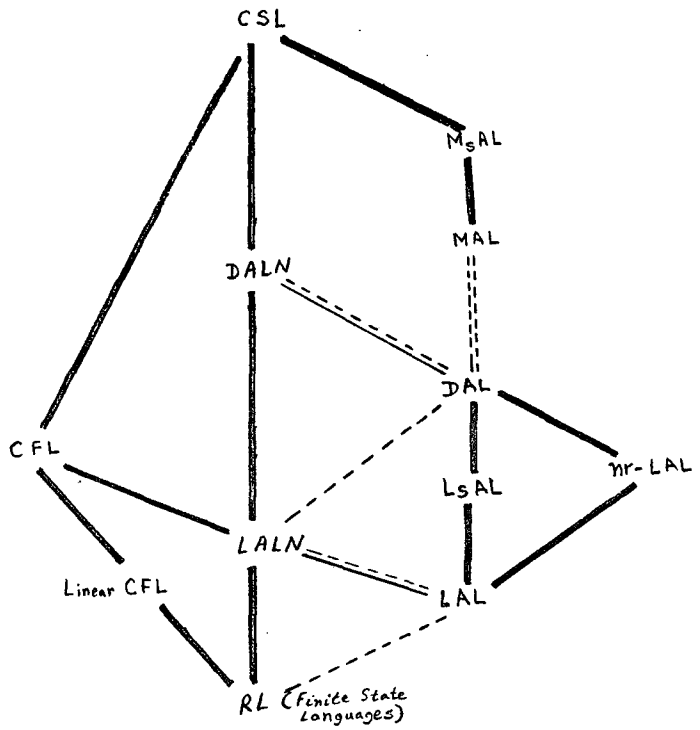
---

* Note that there are very strong restrictions on the repeatability of these primitive adjuncts.

** In principle, we could consider arbitrarily many intermediate stages, between the first and the last MAG's. However, there would not be much point in considering such sequences, unless each intermediate stage has some reasonable linguistic interpretation.

where G" underlies G and G underlies G'. Further development of these ideas in some detail will be reported in Joshi (1969).

Appendix: Fig. A. summarizes the hierarchy of certain subclasses of AL's and MAL's in relation to the phrase structure hierarchy. (The replacement rule in an MAG can be generalized in such a way that all occurences of S in a complex basic string are simultaneously replaced by a specified set of replacer strings. We call such a grammar an MAG with simultaneous replacement rules, M_sAG, and the corresponding language, M_sAL. It can be shown that MAL $\subsetneq$ M_sAL $\subsetneq$ CSL. An M_sAL has the property that the lengths of the strings in it (assuming an ordering in terms of increasing lengths) grow no faster than an exponential. The whole class of M_sAG's as such does not appear to be linguistically relevant.)



Proper Inclusion ($\subsetneq$)

Conjectured Inclusion ($\subseteq$)

Inclusion with Conjectured Equality

Equality

FIG. A.

References

1. N. Chomsky (1963), Formal Properties of Grammars, Handbook of Mathematical Psychology, John Wiley (1963), Ch. 12.

2. N. Chomsky, M. P. Schützenberger, (1963), Algebraic Theory of Contextfree Languages, Computer Programming and Formal Systems, North Holland, Amsterdam.

3. N. Chomsky (1965), Aspects of Theory of Syntax, M.I.T. Press.

4. S. Ginsburg (1966), Mathematical Theory of Context-free Languages, McGraw-Hill.

5. Z. S. Harris (1962), String Analysis of Language Structure, Mouton, The Hague.

6. Z. S. Harris (1964), Elementary Transformations, Transformations and Discourse Analysis Papers, No. 54, University of Pennsylvania.

7. Z. S. Harris (1965), Transformational Theory, Language, Vol. 41.

8. Z. S. Harris (1968), Mathematical Structures of Language, Interscience Publishers.

9. H. Hiż (1967), Computable and Uncomputable Elements of Syntax, Logic, Methodology and Philosophy of Science III, North Holland, Amsterdam.

10. D. Hiż, A. K. Joshi (1967), Transformational Decomposition, Proc. IFIP Int. Conf. on Computational Linguistics, Grenoble.

11. A. K. Joshi (1966), String Representation of Transformations, Transformations and Discourse Analysis Papers, No. 58, University of Pennsylvania.

12. A. K. Joshi (1968), Transformational Analysis by Computer, Proc. NIH Seminar on Computational Linguistics, Bethesda (1966), Published in 1968, U.S. Dept. H.E.W.

13. A. K. Joshi, S. Kosaraju, H. Yamada (1968), String Adjunct Grammars, Transformations and Discourse Analysis Papers, No. 75. (revised February 1969)

14. A. K. Joshi (1969), String Adjunct Grammars and Transformational Grammars, Transformations and Discourse Analysis Papers, No. 75II, University of Pennsylvania (under preparation).

15. E. L. Keenan (1969), A Logical Base for a Transformational Grammar of English, Ph.D. dissertation, University of Pennsylvania.

16. J. J. Robinson (1968), Dependency Structures and Transformational Rules, IBM Watson Research Center, Research Report, July (1968).

17. J. R. Ross (1967), Constraints on Variables in Syntax, Ph.D. dissertation, M.I.T.

18. N. Sager (1967), Syntactic Analysis of Natural Language, Advances in Computers, Academic Press, Vol. 8.

19. M. P. Schützenberger (1961), Some remarks on Chomsky's Context-free Languages, M.I.T. Quarterly Progress Report, October 1961.