INFORMATION AND CONTROL 23, 260–287 (1973)

Structural Aspects of Local Adjunct Languages*

LEON S. LEVY

University of Delaware,

Several open problems concerning local adjunct languages are considered and solved. One of the most interesting (from a linguistic point of view) and difficult (mathematically) open problems was whether or not null symbols can be dispensed without sacrificing the weak generative capacity. This problem is solved and the answer is negative.

Also considered are some problems concerning one-sided grammars, homomorphisms of languages (it is shown that local adjunct languages are not closed under homomorphism), β-linear languages and mixed adjunct grammars.

1. INTRODUCTION

String adjunct grammars and their generalizations have been studied by Joshi, Kosaraju, and Yamada (1972b) and Joshi (1969, 1972) as an alternate means for describing the generation of formal languages and their classification. Adjunction as an operation on trees has also been studied in Joshi and Takahashi (1971), Levy (1971), and Joshi, Levy, and Takahashi (1972c), and, in fact, the operation of tree adjunction alone has more generative capacity than is necessary to obtain the context-free languages.

The class of languages generated by local string adjunct grammars is a subset of the class of context-free languages. One might ask: Since context-free grammars are inadequate for characterizing natural languages, why should one study subclasses of context-free grammars? There are two reasons:

(1) One is interested in obtaining the "simplest" and "most natural" linguistic mechanisms available in natural language, Harris (1961, 1968), and in characterizing these using the methods of the theory of formal languages. In this paper, only the context-free operations are studied, namely local adjunction and simple replacement. Their linguistic relevance

* Partially supported by NSF GS-2509, NSF GS-27, and U.S. Army Research Office, Durham (DA 31-124 ARO(D)-98).

has been discussed in Joshi, Kosaraju, and Yamada (1972b) and Joshi (1969, 1972).

(2) One uses these grammars (e.g., mixed adjunct grammars) as the base component of transformational grammars. We like to narrow down each component of a transformational grammar. String adjunct grammars and mixed adjunct grammars provide a very natural way of narrowing down the base component of a transformational grammar.

The motivation for adjunction is to characterize the notion of a head of a constituent (which is awkward to characterize in a phrase structure grammar). (Example: In *new books from the reference library*, *new* and *from the library* are adjoined to *books*, while *reference* is adjoined to *library*.) Adjunction of strings, defined formally in Section 2, turns out to be a surprisingly powerful operation, but one whose characteristics are quite different from the usual phrase structure descriptions. Thus, the standard closure properties are either difficult to study or turn out to be false. Closure under intersection with regular sets fails, and in Section 5 it is shown that local adjunct languages are not closed under homomorphism. It is not known whether local adjunct languages are closed under union, concatenation, or Kleene star. Joshi, Kosaraju, and Yamada (1972b) in attempting to establish these closure properties introduced a set of special markers, called null symbols, which were used in the grammatical derivations but deleted in the languages and were able to show that with these auxiliary symbols, closure under union, concatenation, and Kleene star obtains. They then conjectured that the null symbols added no generative power to the class of local adjunct grammars. In Section 3, it is shown that the null symbols do, in fact, increase the generative capacity of the local adjunct grammars.

In Section 4 the role of both left and right adjunction is clarified showing that adjoining to the left and right is more powerful than just adjoining to one side. In Section 5, the relationship between local adjunct languages and right linear tree adjunct languages, defined in Joshi, Levy, and Takahashi (1972c), is developed, and the nonclosure of the local adjunct languages under homomorphism is shown.

In Sections 6 and 7, the relationship of the linear context-free languages and the local adjunct languages is studied, using the replacement operator and mixed adjunct grammars of Joshi (1969). (Replacement rules were introduced to take care of exocentric constituents; for example, *whether he came* in *I don't know whether he came*.) The adjunction and replacement operators are quite different in their generative capacities, yielding incomparable subclasses of the class of context-free languages. When combined,

these operators still do not suffice to generate all context-free languages. The class of context-free languages is, however, properly included in both the classes of distributed adjunct languages, Joshi, Kosaraju, and Yamada (1972b) and tree adjunct languages, Joshi, Levy, and Takahashi (1972c). These latter two classes of languages can easily be shown to be incomparable.

2. Definitions for String Adjunct Grammars

Definition 2.1. A *local adjunct grammar with null symbols* (LAGN), G, is (formally described as) a list of seven items, $G = (A, N, \Sigma, \Sigma_c, \Sigma_h, \Sigma_a, J)$, where

A is a finite alphabet;

N is a finite (possibly empty) set of null symbols, such that $A \cap N = \varnothing$;

Σ is a finite set of basic strings over $A \cup N$;

$\Sigma_c \leqslant \Sigma$ is a set of center strings;

$\Sigma_h \leqslant \Sigma$ is a set of host strings;

$\Sigma_a \leqslant \Sigma$ is a set of adjunct strings;

J is a finite set of adjunction rules.

Each adjunction rule u in J is of the form $u = (\sigma_i, \sigma_j, \xi_k)$ where σ_i is a host string, σ_j is an adjunct string, and ξ_k is a point of adjunction; each σ in Σ is in $(A \cup N)^* A(A \cup N)^*$; and each point of adjunction, ξ_k, is to the left or right of a symbol in A, null symbols having no points of adjunction. In writing adjunction rules, each adjunction point will be either l_q or r_q with q designating the qth nonnull symbol counting from the left of the host string. (It was shown by Joshi, Kosaraju and Yamada (1972b) that every LAGN can be put in standard form where each string has at most one null symbol. Further, in such a standard form the null symbol can always be made the leftmost symbol in the string. Henceforth, only standard form LAGN's will be used.)

Remark. A local adjunct grammar (LAG) is an LAGN where $N = \varnothing$.

Example 2.1. The following is an LAGN:

$G = (A, N, \Sigma, \Sigma_c, \Sigma_h, \Sigma_a, J)$;

$A = \{a, b\}$;

$N = \{\alpha, \beta\}$;

$\Sigma = \{\alpha ab, \beta ab\}$,

$\Sigma_c = \{\alpha ab\}$,

$\Sigma_h = \Sigma_a = \Sigma$;

$J = \{(\alpha ab, \beta ab, r_1), (\beta ab, \alpha ab, r_1)\}$.

In defining the language generated by a local adjunct grammar, reference must be made to the *syntactic classes* of strings in Σ. If a string of syntactic class P is adjoined to a string of syntactic class Q, by some rule in J, the result will be in syntactic class Q. Markers are used in the language generation process for identification of syntactic classes, as follows: Let $\hat{A} = \{\hat{a}_i \mid a_i$ is in $A\}$; let $\hat{N} = \{\hat{\nu}_i \mid \nu_i$ is in $N\}$; and let ϵ denote the empty string.

The following homomorphisms are introduced.[1] Each homomorphism is defined in terms of its operation on symbols; the extension to strings and sets of strings are the usual ones: $H(xy) = H(x)\,H(y)$; $H(S_1 \cup S_2) = H(S_1) \cup H(S_2)$.

(i) D is a homomorphism which deletes unmarked symbols and leaves marked symbols without their marks. D is defined for strings over $A \cup \hat{A} \cup N \cup \hat{N}$ by:

$$D(x) = \epsilon \quad \text{for} \quad x \text{ in } A \cup N,$$

$$D(\hat{x}) = x \quad \text{for} \quad \hat{x} \text{ in } \hat{A} \cup \hat{N}.$$

EXAMPLE 2.2. $D(\hat{v}a\hat{a}b\hat{c}) = vac$.

(ii) H is a homomorphism which removes markers and deletes null symbols. H is defined for strings over $A \cup \hat{A} \cup N \cup \hat{N}$.

$$H(x) = \epsilon \quad \text{for} \quad x \text{ in } N \cup \hat{N},$$

$$H(x) = x \quad \text{for} \quad x \text{ in } A,$$

$$H(\hat{x}) = x \quad \text{for} \quad \hat{x} \text{ in } \hat{A}.$$

EXAMPLE 2.3. $H(va\hat{b}) = ab$.

(iii) I is a homomorphism which adds markers. I is defined for strings over $A \cup N$.

$$I(x) = \hat{x} \quad \text{for} \quad x \text{ in } A \cup N.$$

[1] These homomorphism operators were not used in the original local adjunct grammar paper (Joshi, Kosaraju, Yamada, 1972b).

EXAMPLE 2.4. $I(vab) = \hat{v}\hat{a}\hat{b}$.

S is a substitution operator corresponding to adjunction, having three operands, one of which is a subscript on S. In $S_\xi(X, Y)$, X and Y are strings and ξ is either l_q or r_q. $S_\xi(X, Y)$ is the string obtained by adjoining Y to the left or right of the qth symbol of $\hat{A}$ in X as ξ is l_q or r_q, respectively.

EXAMPLE 2.5. $S_{r_2}(\hat{v}\hat{a}b\hat{c}de, fg) = \hat{v}\hat{a}b\hat{c}fgde$.

Now the definition of a local adjunct language with null symbols is as follows.

DEFINITION 2.2. A *local adjunct language with null symbols* (LALN) is the set of strings over A such that each string is derived from some string in Σ_c. Formally, $L = H(\hat{\Sigma}(\Sigma_c))$ where $\hat{\Sigma}$ is defined recursively as:

(i) If σ_i is in Σ, then $I(\sigma_i)$ is in $\hat{\Sigma}(\sigma_i)$.

(ii) If σ_i is in $\hat{\Sigma}(\sigma_j)$ and σ_k is in $\hat{\Sigma}(\sigma_1)$ and $(\sigma_j, \sigma_1, \xi)$ is a rule in J, then

$$S_\xi(\sigma_i, H(\sigma_k)) \qquad \text{is in } \hat{\Sigma}(\sigma_j).$$

(iii) Nothing else is in $\hat{\Sigma}(\sigma_i)$ unless it follows from (i) and (ii).

For any σ in Σ, $\hat{\Sigma}(\sigma)$ is the *syntactic class* of σ. $\hat{\Sigma}(\Sigma_c)$ is the union of the syntactic classes of all strings in Σ_c; i.e., $\hat{\Sigma}(\Sigma_c) = \bigcup_{\sigma \text{in} \Sigma_c} \hat{\Sigma}(\sigma)$.

Note that if σ_i is in $\hat{\Sigma}(\sigma_j)$, then $D(\sigma_i) = \sigma_j$.

A local adjunct language (LAL) is the language derived from an LAG (i.e., $N = \varnothing$).

EXAMPLE 2.6. Let G be the LAGN given in Example 2.1, then $L(G) = \{x \mid x \text{ is in } a(a \vee b)^*b \text{ and which is a balanced parenthesis string when } a \text{ is considered a left parenthesis and } b \text{ considered a right parenthesis}\}$.

Note that the language of Example 2.6 could have been derived from a local adjunct grammar (without null symbols), using the grammar $G = (A, N, \Sigma, \Sigma_c, \Sigma_h, \Sigma_a, J)$ with $A = \{a, b\}$, $N = \varnothing$, $\Sigma = \Sigma_h = \Sigma_c = \Sigma_a = \{ab\}$ and $J = \{(ab, ab, r_1)\}$.

3. ROLE OF THE NULL SYMBOLS

In the LAGN, G, given in Example 2.1, null symbols were used but it was noted, following the description of the derived language, $L(G)$, given in Example 2.6, that an equivalent grammar without null symbols generates

the same language. Indeed, it is often the case that the null symbols serve in a convenient, but not essential, role.

The null symbols were originally introduced to allow proofs that:

(i) the union of two LALN's is an LALN;

(ii) the concatenation of two LALN's is an LALN;

(iii) the Kleene closure (without $\{\epsilon\}$) of an LALN is an LALN.

These facts, together with the observation that every finite set of strings is an LALN, suffice to show that the (ϵ-free) regular sets are LALN's.

The presence of null symbols in strings of the grammar corresponds to different structural descriptions of the *basic sentences* of the language, and one would prefer to have a unique structural description for each *basic sentence* in the language.[2] Joshi *et al.* (1972b) conjectured that for any LAGN one could find an LAG (without null symbols) which was weakly equivalent. This conjecture is refuted by Theorem 3.1, which shows that the class of LAL's is properly contained in the class of LALN's.

The result that null symbols *do* increase the weak generative capacity of the language is established by a diagonalization proof. A grammar, G_0, with null symbols is exhibited, and it is demonstrated that there is no weakly equivalent grammar without null symbols. In somewhat greater detail, a language L' properly contained in $L(G_0)$ is exhibited, such that in any grammar, without null symbols, any rule used to generate a word of L' must satisfy a particular predicate. It is then possible to establish a total ordering of rules satisfying the given predicate. Then, given any finite set of rules of this form, there is an effective construction of a word in L' not generable by the grammar in question.

Let G_0 be the following LAGN:

$$A = \{a, b, c, d, e, f\}$$

$$N = \{\mu\}$$

$$\Sigma = \Sigma_c = \Sigma_h = \Sigma_a = \{ab, \mu ab, cd, \mu cd, ef, \mu ef\}$$

$$J = \text{set of adjunction rules} = \{(ab, cd, r_1), \quad (ab, \mu cd, r_1),$$
$$(ab, ef, l_2), \quad (ab, \mu ef, l_2),$$
$$(cd, cd, r_1), \quad (cd, \mu cd, r_1),$$
$$(cd, ef, l_2), \quad (cd, \mu ef, l_2),$$

[2] Of course, even with unambiguous interpretation of the *basic sentences*, there are still ambiguous local adjunct grammars.

(ef, cd, r_1), $(ef, \mu cd, r_1)$,
(ef, ef, l_2), $(ef, \mu ef, l_2)$,
$(\mu ab, ab, r_1)$, $(\mu ab, \mu ab, r_1)$,
$(\mu ab, cd, l_2)$, $(\mu ab, \mu cd, l_2)$,
$(\mu cd, ab, r_1)$, $(\mu cd, \mu ab, r_1)$,
$(\mu cd, cd, l_2)$, $(\mu cd, \mu cd, l_2)$,
$(\mu ef, ab, r_1)$, $(\mu ef, \mu ab, r_1)$,
$(\mu ef, cd, l_2)$, $(\mu ef, \mu cd, l_2)\}$.

G_0 has six host strings. For each host string there are four adjunction rules. Altogether there are 24 string adjunction rules in the grammar. What follows is a proof that, in G_0, the null symbols play an essential role, since it will be shown that no grammar G_0', without null symbols, can generate $L(G_0)$.

DEFINITION 3.1. Let L'' be the Dyck Language (without null string) defined by considering (a, b), (c, d), and (e, f) to be matching pairs. If x is in L'', then x is said to be *balanced.* If every word in a language is balanced, then the language is balanced.

DEFINITION 3.2. Let $L' = L'' \cap (a \vee c \vee e)^* (b \vee d \vee f)^*$. If x is in L', then we say that x is *linearly nested.*

DEFINITION 3.3. A *preimage* of a word $x = x_1 x_2 \cdots x_n$ is any word in $N^* x_1 N^* x_2 N^* \cdots N^* x_n N^*$.

PROPOSITION 3.1. $L(G_0)$ *is balanced.*

Proof. Each string in Σ is a preimage of a balanced word and each string formed by adjoining balanced strings to the preimages of balanced strings is the preimage of a balanced string. Thus, under the homomorphism, H, which erases null symbols, every string in $L(G_0)$ is balanced. Q.E.D.

LEMMA 3.1. $L' \subset L(G_0)$. (*Every linearly nested word is in* $L(G_0)$.)

Proof. Let h be the homomorphism, h: $a \mapsto b$, $c \mapsto d$, $e \mapsto f$, extended to strings over $\{a, c, e\}$, as usual. The lemma claims that among the linearly nested words in $L(G_0)$ can be found any word of the form xy^R where x is in $(a \vee c \vee e)^*$, $y = h(x)$, and y^R is the mirror image of y. Proof is by induction on $| x |$. If $| x | = 1$, then a preimage of $x[h(x)]^R$ is generable

in one of the syntactic classes of G_0, since $x[h(x)]^R$ is a preimage of itself and is in Σ_c. Suppose that for $|x| = k$, a preimage of $x[h(x)]^R$ is in the syntactic class of $\alpha qh(q)$, where q is in A and α is in $N \cup \{\epsilon\}$. Then for any p in A one can find a β in $N \cup \{\epsilon\}$ such that a preimage of $px[h(px)]^R$ is in the syntactic class of $\beta ph(p)$. This is so because the string adjunction rules of G_0 have been so constructed. (In particular, if $q = a$ or $q = c$, choose $\beta = \mu$; if $q = e$, choose $\beta =$ the empty string). Q.E.D.

The proof that no local adjunct grammar, G_0' (without null symbols) can generate $L(G_0)$ is developed by considering the set of rules necessary to generate all words in L' while generating only words in $L(G_0)$. It will be shown that given any local adjunct grammar, G_0', without null symbols, such that $L(G_0') \subseteq L(G_0)$, a string x can be effectively constructed so that x is in $L' \cap \overline{L(G_0')}$.

DEFINITION 3.4. A rule u is said to be *admissible* iff $u = (\sigma_i, \sigma_j, \xi_k)$ implies that σ_i is linearly nested, σ_j is linearly nested, and if $\xi_k = r_n$ then[3] $n = 1/2\,|\,\sigma_i\,|$ while if $\xi_k = l_n$ then $n = 1/2\,|\,\sigma_i\,| + 1$, where $|\,\sigma_i\,|$ is the length of σ_i. (Informally, a rule is admissible iff the host and adjunct strings are linearly nested, and adjunction occurs "in the middle of the string.")

LEMMA 3.2. *Any derivation in G_0' of any word in L' uses only admissible rules.*

Proof. Each host or adjunct string, σ, used in derivation of any word $x = x_1x_2 \cdots x_n$ is of the form

$$\sigma = x_{i_1}x_{i_2} \cdots x_{i_k} \qquad \text{where} \quad 1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant n.$$

Thus, x in $(a \vee c \vee e)^*(b \vee d \vee f)^*$ implies $\sigma = \sigma_1\sigma_2$ is in $(a \vee c \vee e)^*(b \vee d \vee f)^*$, where σ_1 is in $(a \vee c \vee e)^*$ and σ_2 is in $(b \vee d \vee f)^*$. If σ is not linearly nested, then either $|\,\sigma_1\,| \neq |\,\sigma_2\,|$ or $|\,\sigma_1\,| = |\,\sigma_2\,|$ but the symbols are not in symmetrically located positions for balancing. If $|\,\sigma_1\,| \neq |\,\sigma_2\,|$ then it is easy to generate unbalanced strings, hence, by Proposition 3.1, words not in $L(G_0)$. Otherwise, $|\,\sigma_1\,| = |\,\sigma_2\,|$ and the symbols are not symmetrically positioned, so if σ is a center string, it is not in $L(G_0)$, while if σ is an adjunct string, any word formed immediately by its adjunction is not in $L(G_0)$.

Finally, if $\sigma_1 = [h(\sigma_2)]^R$, and a word in L' is being generated, adjunction must occur in the middle to preserve the linear nested property. Q.E.D.

[3] $|\,x\,|$ denotes the length of the string x.

Let S be the set of admissible rules in G_0'.

DEFINITION 3.5. (Ordering on strings). Let $\sigma_i <_p \sigma_j$ if

(i) $|\sigma_i| < |\sigma_j|$

or

(ii) $|\sigma_i| = |\sigma_j|$ and σ_i lexicographically precedes σ_j.

$$\sigma_i \leqslant_p \sigma_j \quad \text{if} \quad \sigma_i <_p \sigma_j \quad \text{or} \quad \sigma_i = \sigma_j .$$

FACT 3.1. *$<_p$ is a total ordering on strings. We extend $<_p$ to rules in G_0'.*

DEFINITION 3.6 (Ordering on rules). $u_i <_p u_j$ if

(i) $\text{host}(u_i) <_p \text{host}(u_j)$ or

(ii) $\text{host}(u_i) = \text{host}(u_j)$ and $\text{adjunct}(u_i) <_p \text{adjunct}(u_j)$ or

(iii) $\text{host}(u_i) = \text{host}(u_j)$ and $\text{adjunct}(u_i) = \text{adjunct}(u_j)$ and $|\xi_i| < |\xi_j|$, where $|\xi| = q$ if $\xi = r_q$ or l_q.

FACT 3.2. *$<_p$ is a total ordering on rules in S.*

Remark. In rules in G_0', there might be a pair of rules $(\sigma_1, \sigma_2, l_q)$, $(\sigma_1, \sigma_2, r_q)$ which are not ordered by $<_p$. However in S, q is either $1/2\,|\sigma_1|$ if ξ is r_q, or $1/2\,|\sigma_1| + 1$ if ξ is l_q.

DEFINITION 3.7. $hd(\sigma) = \{x \mid x \text{ is in } A \text{ and } H(\sigma) = x\sigma'\}$. ($hd(\sigma)$ is the first nonnull symbol of σ.)

DEFINITION 3.8. $n(\sigma) = \{y \mid (\exists u \text{ in } S)(\sigma = \text{host}(u) \,\&\, y = hd(\text{adjunct}(u))\}$.

LEMMA 3.3. *For any σ in G_0', $|n(\sigma)| \leqslant 2$.*

Proof. Let L be a language. We say that L has *property A*, if for any $x = x_1 s_1 s_2 s_3 x_2$ in L with s_1, s_2, s_3 balanced, either $hd(s_1) = hd(s_2)$ or $hd(s_2) = hd(s_3)$.

Recall that G_0' is such that $L(G_0') \subseteq L(G_0)$. In G_0, each derivation consists of a sequence of adjunctions of balanced words to preimages of balanced words. Now for each σ in G_0, $|n(\sigma)| = 2$. Also, the rules in G_0 are such that if $x_1 x_2$ is in syntactic class P and adjunction can be made between x_1 and x_2 of strings in $(a_1 A^* \vee a_2 A^*)$, then a sequence of adjunctions can

yield words in $x_1(\Psi_1)^*(\Psi_2)^* x_2$ or words in $x_1(\Psi_2)^*(\Psi_1)^* x_2$, where $\Psi_i = \{\sigma \mid hd(\sigma) = a_i \text{ and } \sigma \text{ is balanced}\}$. Thus, $L(G_0)$ has property A.

Since $L(G_0') \subseteq L(G_0)$, $L(G_0')$ has property A. But if $|n(\sigma)| = 3$ in G_0', we can generate words in $L(G_0')$ of the form $x_1 s_1 s_2 s_3 x_2$ such that $hd(s_1) \neq hd(s_2)$ and $hd(s_2) \neq hd(s_3)$. Thus, it must be that $|n(\sigma)| \leqslant 2$. Q.E.D.

DEFINITION 3.9. $m(\sigma) = \{x \mid x \text{ is in } \{a, c, e\}, x \text{ is not in } n(\sigma), \& (\forall y)(y \text{ not in } n(\sigma) \text{ implies } x \leqslant y)\}$. That is, $m(\sigma)$ is the least y in $\{a, c, e\}$ which cannot be the head of an adjunct to σ in the rules in S.

LEMMA 3.4. *Given any local adjunct grammar, G_0', without null symbols, such that $L(G_0') \subseteq L(G_0)$, then $L(G_0') \neq L(G_0)$.*

Proof. From the description of G_0', an x in L' which is not in $L(G_0')$ can be constructed. First note that by Lemma 3.2, only the set of admissible rules, S, need be considered. Also, the rules in S may be ordered by the total ordering, $<_p$, as $u_1, u_2, \ldots, u_k$. (Fact 3.2).

Let $\text{host}(u_i) \underset{\Delta}{=} V_i \underset{\Delta}{=} p(u_i)\, s(u_i)$ where $|p(u_i)| = |s(u_i)|$. Define

$$\sigma^1 = p(u_1),$$

$$\tau^1 = \sigma^1 m(V_1),$$

$$v^{r+1} = \{u_j \mid (\exists x)(V_j = \tau^r x \text{ and } (\forall k)(\exists x)(V_k = \tau^r x \text{ implies } j \leqslant k)\},$$

$$u^{r+1} = \begin{cases} \text{the unique member of } v^{r+1} \text{ if } v^{r+1} \text{ is nonempty,} \\ \text{undefined otherwise,} \end{cases}$$

where

$$\sigma^{r+1} = \begin{cases} p(u^{r+1}) \text{ if } v^{r+1} \neq \varnothing, \\ \text{undefined otherwise,} \end{cases}$$

and

$$\tau^{r+1} = \begin{cases} \sigma^{r+1} m(\text{host}(u^{r+1})) \text{ if } v^{r+1} \neq \varnothing, \\ \text{undefined otherwise.} \end{cases}$$

The preceding recursive construction of σ^r, τ^r, u^r, v^r starts with u_1 and finds τ^1 as the prefix of some string in L' which cannot be generated with u_1 as the last applied rule but must be generated by a later rule u^2 in which $\text{host}(u^2)$ is of the form $\tau^1 x$. For such a rule, there will be a string with τ^2 as prefix which is not generable by u^2, or any rule $<_p u^2$, as the last applied rule. Continuing in this way, v^{r+1} is the singleton set containing the least

rule in which the host has a prefix $\sigma^r m(\text{host}(u^r))$ if such a rule exists; otherwise v^{r+1} is empty.

Now, let $\hat{\sigma} = \{\tau^M x[h(\tau^M x)]^R \mid M$ is the largest index for which τ^r is defined and x in $(a \vee c \vee e)^*$ chosen so that $|\hat{\sigma}| >$ longest string in $\Sigma_0\}$.

By construction τ^M cannot be the prefix of any string generated by the rules in S. Yet $\hat{\sigma}$ is in L'. Thus, $\hat{\sigma}$ is the desired word whose construction has been asserted; i.e., $\hat{\sigma}$ is in L' but $\hat{\sigma}$ is not in $L(G_0')$. Therefore, $L(G_0') \neq L(G_0)$. Q.E.D.

From Lemma 3.4, we immediately have the following theorem.

THEOREM 3.1. LAL $\subsetneq$ LALN. (*Null symbols increase the weak generative capacity of local adjunct grammars.*)

Theorem 3.1 while establishing a role for null symbols leaves the following.

Open question. Is every regular set an LAL; i.e., can every regular set be generated without the use of null symbols?

Also of interest are the questions of whether the LAL's are closed under union, concatenation, or Kleene star.

4. ONE-SIDED LAGN'S

It is often the case that we can construct an adjunct grammar for a given language which requires only one-sided local adjunction (in which case there is no distinction between left and right). In this section it is proved that LAGN's with both left and right adjunction are more powerful than LAGN's with one-sided adjunction only. These results carry over to the case of mixed adjunct grammars, discussed in Section 7. Several definitions are required.

DEFINITION 4.1 (Joshi *et al.* (1972b)). A *derivation tree* has a set of nodes, each node labeled with a string in Σ and defined recursively as follows:

(i) A single node labeled by a string in Σ is a derivation tree.

(ii) If T is a derivation tree, then T' is a derivation tree if T' is obtained from T by adding a node labeled σ, and a branch labeled ξ_k, directed from the node labeled σ to some node in T.

(iii) Nothing else.

T is a derivation tree in G if each branch in T corresponds to a rule in G;

i.e., if σ_i and σ_j are labeled nodes and ξ_k is a labeled edge from σ_j to σ_i, then $(\sigma_i, \sigma_j, \xi_k)$ is a rule in G. The convention is adopted that all edges directed to a vertex are ordered so that if an edge from σ_1 labeled ξ_1 occurs to the right of an edge from σ_2 labeled ξ_2, then either $\xi_1 = \xi_2$ or ξ_2 corresponds to a point of adjunction to the left of ξ_1. Moreover, when $\xi_j = \xi_k$ and if they are both left adjunction points, and if ξ_j labels a point to the right of ξ_k, then the string derived at ξ_j is adjoined before the string derived at ξ_k. Dually, when $\xi_j = \xi_k$ and both are right adjunction points, and ξ_j labels a branch to the left of ξ_k, the string derived at ξ_j is adjoined first.

Finally, T represents the derivation of a word in $L(G)$ if T is a derivation tree representing a derivation in G and the root of T is in Σ_c.

EXAMPLE 4.1.

$$A = \{a, b\}, \qquad N = \varnothing,$$
$$\Sigma = \Sigma_c = \Sigma_h = \Sigma_a = \{ab, aa, ba\},$$
$$J = \{(ab, aa, l_1), (ab, ba, r_1), (ab, ab, r_1), (ba, ba, r_2), (aa, aa, r_2)\}.$$

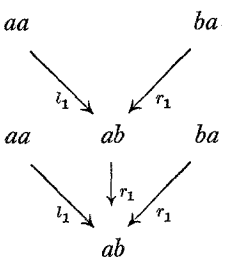


FIG. 1. Derivation tree for *aaaaaababbab*. (The string built up at the interior node is *aaabab*.) Fig. 1 follows Example 4.1.

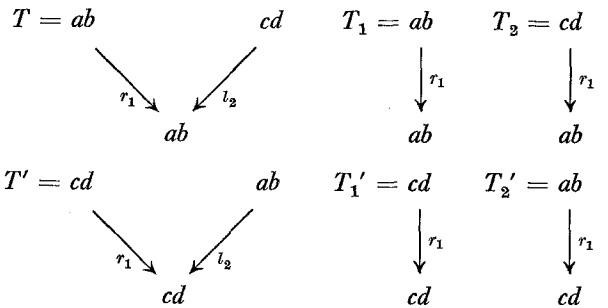
DEFINITION 4.2. Let T, T' be derivation trees in G, an *occurrence* of T in T' is a tree T obtained from T' by deleting (0 or more) branches and nodes of T'. T is said to occur at a node x in T' if T occurs in T' and the root of T is x. The string derived by T will be denoted $\sigma(T)$.

DEFINITION 4.3. If σ_1 and σ_2 are strings, an *occurrence* of σ_1 in σ_2 is a string $H(\sigma_1)$ obtained from σ_2 by deleting (0 or more) symbols of σ_2. Note that null symbols of σ_1 need not occur and that the portions of $H(\sigma_1)$ occurring in σ_2 need not be contiguous.

EXAMPLE 4.2. $\sigma_1 = vabc$; $\sigma_2 = \mu adbecc$. There are two occurrences of σ_1 in σ_2 as underlined: $\mu\underline{a}d\underline{b}e\underline{c}c$, $\mu\underline{a}d\underline{b}ec\underline{c}$.

A string σ occurring in a derivation d in G' means that σ occurs in some basic string σ' in G' and that σ' labels a node of the derivation tree for d.

Let



and let G be the following LAG:

$$A = \{a, b, c, d\},$$

$$\Sigma = \Sigma_c = \Sigma_h = \Sigma_a = \{ab, cd\},$$

$$J = \{(ab, ab, r_1), (ab, cd, l_2), (cd, ab, l_2), (cd, cd, r_1)\}.$$

LEMMA 4.1. *Let G be as given above, and let G' be any one-sided* LAGN *such that $L(G') = L(G)$. Let l be the length of the longest center string of G'. Whenever, in a derivation of z in G, such that $|z| > l$, the subtree T occurs, then in G', $\sigma(T_1)$ or $\sigma(T_2)$ must occur in a string in the grammar G', in any derivation of z. Similarly, whenever T' occurs in a derivation, $\sigma(T_1')$ or $\sigma(T_2')$ must occur.*

Proof. Assume that T occurs in G but neither $\sigma(T_1)$ nor $\sigma(T_2)$ occur in a corresponding derivation in G'. Then the occurrence of $\sigma(T_1)$ and $\sigma(T_2)$ in derived strings in $L(G')$ must be the result of adjunction. This could occur only if (i) strings of the form *axb* and *cyd* can both be adjoined at the same point, or (ii) a string of one of these forms can be adjoined and act as a host for the other. Either of these possibilities is easily seen to lead to the generation of a string not in $L(G)$. Q.E.D.

LEMMA 4.2. *In the set of derivation trees of depth $k > l = \max\{|\sigma| \mid \sigma$ in $\Sigma_c\}$ over G, there is a set of trees $\{T_i\}$ each T_i being of depth k, such that at least one of $\{\sigma(T_i)\}$ must occur in a string in any one-sided grammar for $L(G)$.*

Proof. Call a tree which at each node except for the terminal nodes has an occurrence of T or T', a *complete tree.* A tree, all of whose terminal

nodes are at the same depth, is called a *uniform tree*. Let T_A be a tree in G and T_B a tree in G' such that $\sigma(T_A) = \sigma(T_B)$. We claim that for each complete tree T_A there is an occurrence in T_B of $\sigma(T_C)$ where T_C is a tree which occurs at the root of T_A, and has, as one of its leaves, a leaf of T_A.

By the preceding lemma, for derivation of words of length $>l$, $\sigma(T_1)$ or $\sigma(T_2)$ must occur in the root label of $\sigma(T_B)$. Assume that T_A is a uniform complete tree of depth $k' > k$ and assume that there is in the root label of T_B an occurrence of $\sigma(T_C)$ of a tree T_C of depth k at most, where T_C is a subtree of T_A occurring at the root of T_A. Let n be a terminal node of T_C at depth k and let T_D be the subtree of T_A occurring at n. Now, either T or T' occurs at the root of T_D, and both *axb* and *cyd* cannot be adjoined in G', which is a one-sided adjunct grammar, to the same point of $\sigma(T_C)$, or we could generate a string not in $L(G)$. Thus, at least one of these must occur in the root label of T_B. Thus, the tree whose derived string occurs in the root label of T_B must have, as one of its leaves, a leaf of T_A. Q.E.D.

THEOREM 4.1. *There is an* LAL *which is not generable by any one-sided* LALN.

Proof. By the preceding lemma, there is no fixed upper bound on the length of a string in a one-sided grammar for $L(G)$. Hence, no one-sided LAGN. Q.E.D.

5. HOMOMORPHISM OF LALN'S

In this section it is shown that the LALN's are not closed under homomorphism. The question of closure under homomorphism is left open by Joshi *et al.* (1972b). The proof is by way of connection with tree adjunct grammars.

Tree adjunct grammars were defined by Joshi and Takahashi (1971) and some related results are given in Levy (1971). The discussion here corresponds most closely to a slightly different development given in Joshi, Levy, and Takahashi (1972c).

DEFINITION 5.1. A tree adjunct grammar, G_τ, consists of a set of center strings, τ_C, and a set of adjunct trees, τ_A. Each center tree has root labeled S, each interior node is labeled with a nonterminal symbol, and each leaf is labeled with a terminal symbol or ϵ (denoting the empty string). Each adjunct tree has all leaves except one labeled with terminals or ϵ, and one

leaf labeled with the same nonterminal as the root label; all interior nodes are labeled with nonterminals.

DEFINITION 5.2. The set of derivation trees in G_τ is defined recursively as follows:

(i) Every center tree is a derivation tree.

(ii) If T is a derivation tree and T' is obtained from T by detaching the subtree T'' at some internal node ν, labeled V, attaching a V-tree in τ_A to ν, and attaching T'' to the V-node on the frontier of T'', then T' is a derivation tree.

(iii) Nothing else.

EXAMPLE 5.1. The adjunction of an adjunct tree to a derivation tree to obtain a new derivation tree is shown in Fig. 2. T' is adjoined to T, as described above, yielding T''.

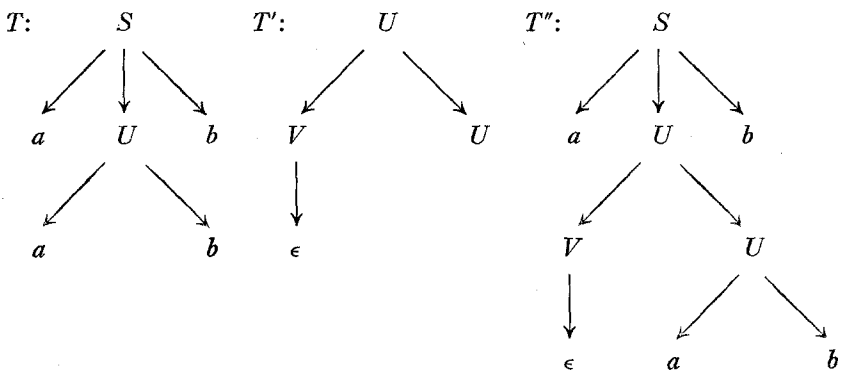


FIG. 2. Tree adjunction (see text).

DEFINITION 5.3. A tree is *linear* if at each level there is at most a single nonterminal. A right-linear (left-linear) tree is a linear tree in which the nonterminal at any level is the rightmost (leftmost) symbol at that level. (A tree is one-sided linear if it is either left-linear or right-linear.) These definitions of right linearity, left linearity, and one-sided linearity are extended to tree adjunct grammars in the usual way.

A language is a right (one-sided) linear tree adjunct language if it is the set of yields of the derivation trees in some right (one-sided) linear grammar.

PROPOSITION 5.1. *Linear context-free and one-sided linear tree adjunct languages are incomparable.*

THEOREM 5.1. *Every* LALN *is a right linear tree adjunct language.*

Proof. We give a construction for the tree adjunct grammar corresponding to a given LALN.

(a) For each center string $\sigma_i = \nu_i x_{i_1} x_{i_2} \cdots x_{i_k}$ add to τ_C the tree shown in Fig. 3a.

(b) For each adjunct string $\sigma_j = \nu_j x_{j_1} x_{j_2} \cdots x_{j_k}$ construct the tree shown in Fig. 3b.

Add to τ_A, for each point of adjunction ν to which it applies, the tree shown in Fig. 3c. Q.E.D.

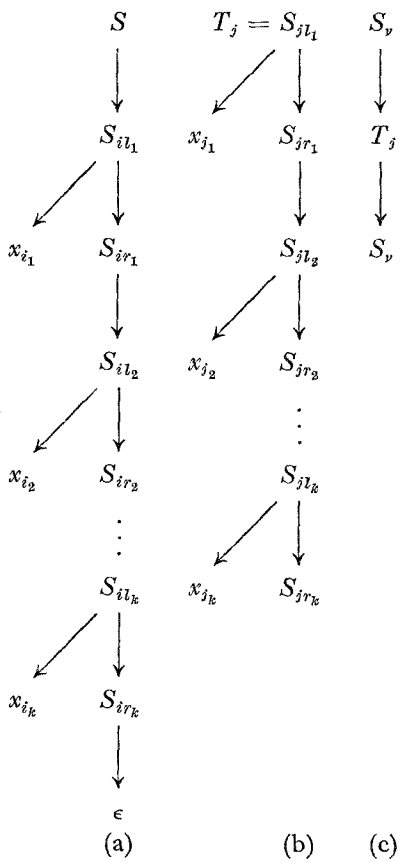


FIG. 3. Trees used in the construction of Theorem 5.1.

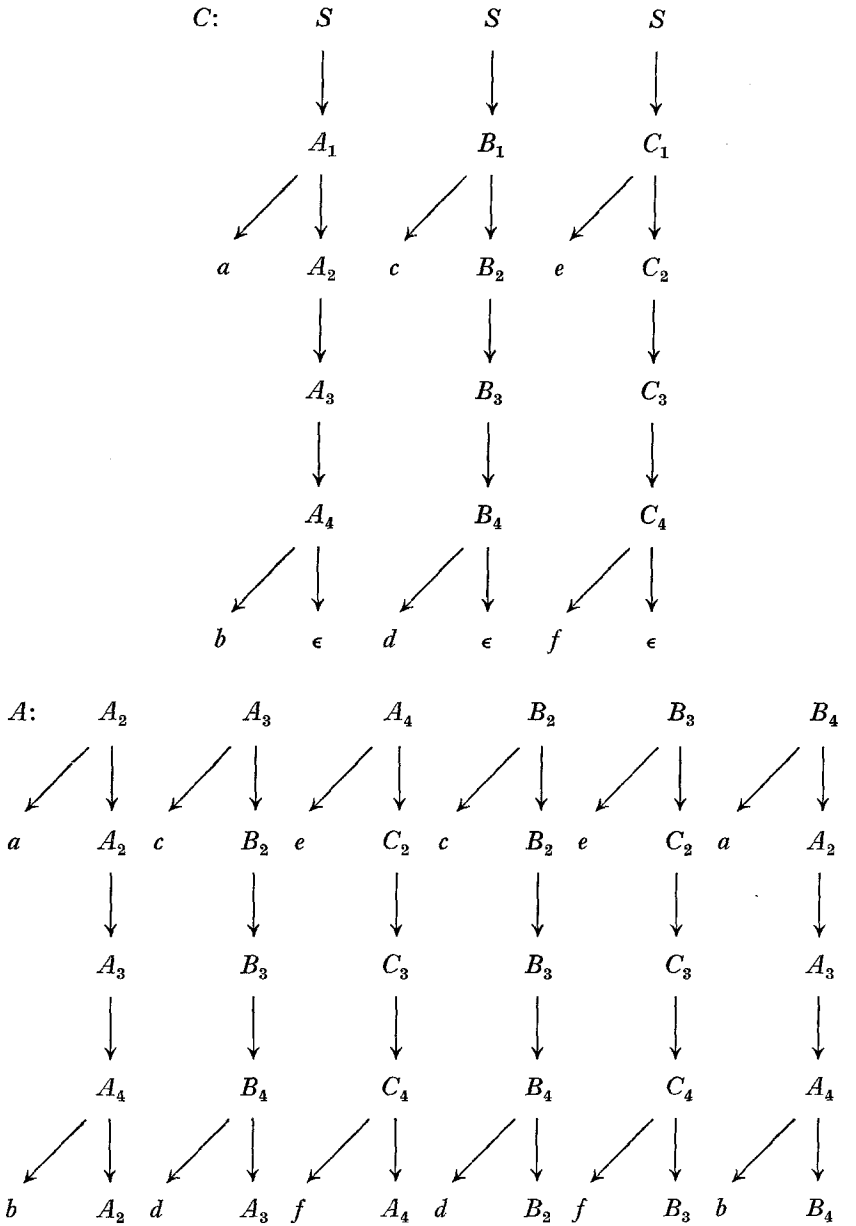


FIG. 4. Tree adjunct grammar of Example 5.2.

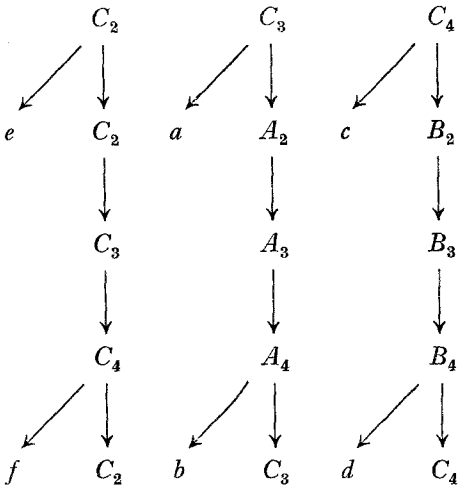
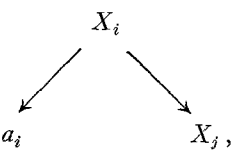


FIG. 4 (*continued*)

LEMMA 5.1. *If* LALN's *were closed under homomorphism, then the class of right linear tree adjunct languages would equal the class of* LALN's.

Proof. Modify the tree adjunct grammar so that at each level, whenever

both X_i and X_j being nonterminals, we have

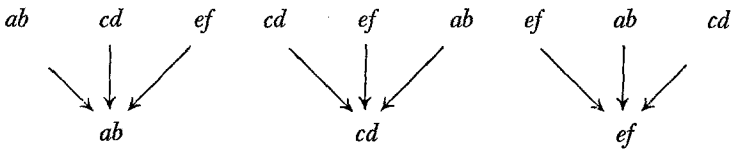


where a_i is a new terminal. The set of yields of the set of derivation trees of the modified tree adjunct grammar is an LALN, 'L', (the grammar can easily be written), and the (unmodified) tree adjunct language is a homomorphic image of L'. Q.E.D.

Rosa Hwang (1972) has constructed the following example.

EXAMPLE 5.2. The tree adjunct grammar shown in Fig. 4 generates a language for which there is no LAGN.

The proof that the grammar given in Example 5.2 yields a language which is not an LALN is similar to the proof of Lemmas 4.1 and 4.2, except that one considers a complete uniform set of trees where at each node there is an occurrence of one of the following three trees:



We omit the details here.

Based on Lemma 5.1 and Example 5.2, we have the following.

THEOREM 5.2. LALN's *are not closed under homomorphism.*

Remark. The local adjunct languages (without null symbols) are also not closed under homomorphism. This can be seen, since it is easy to show that every LALN is a homomorphic image of a one-sided local adjunct language.

6. β-LINEAR LALN'S

In Joshi *et al.* (1972b) the relationship between the linear context-free languages, and the LALN's was explored. It was originally conjectured that any language which is both an LALN and a linear context-free language must be regular. Recently, Hart (1973) has shown an example of a nonregular linear context-free language which is an LALN.

In this section, a class of β-linear LALN's is defined, and it is shown that if L is a β-linear LALN then L is regular.

DEFINITION 6.1 (Rosenberg (1967)). A *β-linear language* is one which can be generated by a grammar in which all productions are of the forms $V \to TV$, $V \to VT$, or $V \to \beta$ where V is any nonterminal, T is any terminal in Σ, and β is a distinguished terminal not in Σ. (It is a linear language generated by a grammar in which all productions whose right side is a single terminal have β as the right side of the productions.) A β-linear LALN is a β-linear language which is an LALN.

DEFINITION 6.2. A *homomorphic mapping of languages* is a mapping, h, from languages to languages which satisfies the condition $h(x) \cup h(y) = h(x \cup y)$ for any languages x, y.

EXAMPLE 6.1. Let $f(L)$ be the operation which removes all words x such that $| x | \leqslant 5$ from a language (if they were in it). It is shown in Levy (1970) that f preserves LALN's and if L_1 and L_2 are LALN's then $(L_1 \cup L_2)$ is an LALN. Finally, $f(L_1) \cup f(L_2) = f(L_1 \cup L_2)$; thus, f is a homomorphic mapping of languages which preserves LALN's.

THEOREM 6.1. *Let L be a* β-LALN. *(L is of the form $\bigcup_i L_i \beta L_i'$ where the L_i and L_i' are languages over Σ, and β is not in Σ.) Let k_1 and k_2 be homomorphic mappings of languages which preserve* LALN's. *Then $\bigcup_i k_1(L_i) \beta k_2(L_i')$ is an* LALN.

Proof. L is derived from some finite set of center strings each of which contains β. Adjunctions to the right of β do not affect anything to the left of β and vice versa (i.e., we can use "separate grammars" with distinguishing null symbols for the adjuncts on each side). Hence, we are free to modify the left and right languages independently. Q.E.D.

THEOREM 6.2. *Let L be a β-linear language, expressible as a finite union in the form $\bigcup_i x_i \beta y_i$, and let k_1 and k_2 be homomorphic mappings of languages which preserve regular sets. Then $L' = \bigcup_i k_1(x_i) \beta k_2(y_i)$ is a β-linear language.*

Proof. In Rosenberg (1967) it is noted that "the class of β-linear languages is coextensive with the class of β-CFL's (a β-CFL is a context-free language $L \subseteq \Sigma^* \beta \Sigma^*$) such that both $\{x \mid (\exists y)(x \beta y \text{ is in } L)\}$ and $\{y \mid (x)(x \beta y \text{ is in } L)\}$ are regular sets."

Now suppose $\bigcup_i L_i \beta L_i'$ is a β-linear language (not necessarily an LALN) and let k_1, k_2 be homomorphic mappings which preserve regular sets, then $\bigcup_i k_1(L_i) k_2(L_i')$ will be β-linear since it preserves both conditions. Q.E.D.

We also have the following corollary to Theorem 6.2.

COROLLARY 6.1. *Let L be a β-linear* LALN. *(L is of the form $\bigcup_i L_i \beta L_i'$ where L_i and L_i' are languages over Σ and β is not in Σ.) Let k_1 and k_2 be homomorphic mappings of languages which preserve regular sets and* LALN's. *Then $\bigcup_i k_1(L_i) \beta k_2(L_i')$ is a β-linear* LALN.

DEFINITION 6.3. If L is a β-linear language, then x such that $(\exists y)(x \beta y$ is in $L)$ is called a *left part*, and $\{y \mid x \beta y$ is in $L\}$ denoted $r(x)$ is called its

corresponding right set. Similarly, *right part* and *corresponding left set* are defined in the obvious way.

THEOREM 6.3. *If L is a β-linear language, then for each left (right) part, the corresponding right (left) set is regular.*

Proof. We give the proof for a left part and its corresponding right set: The rules for a grammar, G, for L are all of the forms: $S_i \to yS_j$; $S_i \to S_j y$; $S_i \to \beta$, where y is any member of V_T, and the start symbol is S_0. If we fix x, we can construct a new β-linear grammar, G', which generates $L(G') = x\beta r(x)$. Since all the rules of G' will be right-linear, $L(G')$ will be regular.

Let $x = x_1 x_2 \cdots x_m$. The construction of G' is as follows:

$$G' = (V_N', V_T', P', S_0^{(0)}),$$

$$V_T' = V_T; \qquad V_N' = \{S_i^{(k)} \mid S_i \text{ is in } V_N \text{ and } 0 \leqslant k \leqslant |x|\},$$

$$P' = \{S_i^{(k-1)} \to x_k S_j^{(k)} \mid S_i \to x_k S_j \text{ is in } G\},$$

$$\cup \{S_i^{(k)} \to S_j^{(k)} y \mid S_i \to S_j y,\ y \text{ in } V_T, \text{ is in } G\},$$

$$\cup \{S_i^{(m)} \to \beta\}.$$

Clearly, $L(G') = x\beta r(x)$. Q.E.D.

EXAMPLE 6.2. Let the productions of G be: $\{S_0 \to aS_1$; $S_1 \to S_2 b$; $S_2 \to aS_1$; $S_2 \to S_2 c$; $S_2 \to dS_3$; $S_3 \to dS_3$; $S_3 \to \beta\}$. The β-linear language with add as its only left part is generated by a grammar with productions: $\{S_0 \to aS_1^{(1)}$; $S_1^{(1)} \to S_2^{(1)} b$; $S_2^{(1)} \to S_2^{(1)} c$; $S_2^{(1)} \to dS_3^{(2)}$; $S_3^{(2)} \to dS_3^{(3)}$; $S_3^{(3)} \to \beta\}$.

COROLLARY 6.2. *Let L be a β-linear language and x be any finite set of left (right) parts of L. Any Boolean function of the corresponding right (left) sets is regular.*

Proof. This is a direct consequence of Theorem 6.3 and the closure of the regular sets under Boolean operations. Q.E.D.

THEOREM 6.4. *Every β-linear* LALN *is regular.*

The proof of Theorem 6.4 is given in the appendix. The proof is essentially constructive, and an example of its application is also given.

7. Mixed Adjunct Grammars

It has been shown in Section 6 that no nonregular β-linear language is an LALN. Also, it was pointed out that many linear languages which are not regular sets are non LALN's, such as $\{a^n b^n \mid n \geqslant 1\}$. Also in Levy (1970) the following is proved.

Theorem 7.1. *Suppose that for some nonnull x, y in A^*, $ux^n vy^n w$ is in L for every $n \geqslant 0$, and v in A^* is nonnull and contains some symbol which appears neither in x nor in y, and for arbitrarily large k', there is an $m > k'$ such that $ux^m vy^{m-p} w$, $1 \leqslant p \leqslant k'$, is not in L, then L is not an* LALN.

As a direct application of Theorem 7.1 it can be shown that the grammar, G, for arithmetic expressions in an ALGOL-like language yields a language which is not an LALN:

$$G = (V_N, V_T, P, S),$$
$$V_N = \{S, L, T\}; \qquad V_T = \{a, b, +, -, \times, /\},$$
$$P = \{S \to S + T;\ S \to T - S;\ S \to T;\ T \to L \times T;$$
$$T \to L/T;\ T \to L;\ L \to (S);\ L \to a;\ L \to b\}.$$

The proof that G is not an LALN is by considering $\{({}^n a)^n \mid n \geqslant 0\} \leqslant L(G)$ and applying Theorem 7.1.

To add to local adjunct grammars the ability to nest parentheses one is led to the use of *replacement rules*. Replacement rules were introduced by Joshi (1969) to account for phrases which are not endocentric and, hence, cannot be built up by adjunction.

Definition 7.1. A *mixed adjunct grammar with null symbols* (MAGN) is (formally described as) a list of nine items,

$$G = (A, N, \Sigma, \Sigma_c, \Sigma_h, \Sigma_a, \Sigma_r, J, R),$$

where

A is a finite alphabet;

N is a finite (possibly empty) set of null symbols;

Σ is a finite set of basic strings over $A \cup N \cup \{S\}$;

$\Sigma_c \leqslant \Sigma$ is the set of center strings;

$\Sigma_h \leqslant \Sigma$ is the set of host strings;

$\Sigma_a \leqslant \Sigma$ is the set of adjunct strings;

$\Sigma_r \leqslant \Sigma$ is the set of replacer strings;

J is a finite set of adjunction rules;

R is a finite set of replacement rules.

Each rule in J is of the form $u = (\sigma_i, \sigma_j, \xi)$, where σ_i is a host string, σ_j is an adjunct string, and ξ is a point of adjunction; each point of adjunction being to the left or right of a symbol in A.

Each rule in R is of the form $r = \langle \sigma_i, \sigma_j \rangle$ where σ_i is in $\Sigma_h \cap \Sigma_S$ and σ_j is in Σ_r. (Σ_S is the set of strings over $A \cup N \cup \{S\}$ containing exactly one occurrence of S.)

EXAMPLE 7.1. An MAGN combining replacement and adjunction rules is

$$\begin{aligned} A &= \{a, b, c, d, e\} \qquad N = \varnothing, \\ \Sigma &= \{aSb, c, de\}, \\ \Sigma_c &= \{de\}, \\ \Sigma_h &= \{aSb, de\}, \\ \Sigma_a &= \Sigma_r = \{aSb, c\}, \\ J &= \{(de, aSb, 1_2)\}, \\ R &= \{\langle aSb, aSb\rangle, \langle aSb, c\rangle\}. \end{aligned}$$

The definition of the language generated by G is, informally, the set of strings derivable from Σ_c, using rules of J and R, with the provision that whenever a string containing an S is to be used either as a replacer or as an adjunct, it must be completely built up (i.e., have no occurrence of S, and have all its adjuncts.)

In order to give the formal definition of the language generated by an MAGN, we must extend some of the homomorphisms, given in Section 2, to the symbol S, and define a substitution operation for replacement.

DEFINITION 7.2. I is a homomorphism which adds markers. I is defined for strings over $A \cup N \cup \{S\}$

$$I(x) = \hat{x} \qquad \text{for} \quad x \text{ in } A \cup N; \qquad I(S) = S.$$

EXAMPLE 7.2. $I(vabS) = \hat{v}\hat{a}\hat{b}S$.

DEFINITION 7.3. R is a replacement operator. The first operand of R is a string having a single occurrence of S, and the second operand is a string over A. $R(X, Y)$ is the string obtained by substituting Y for S in X.

EXAMPLE 7.3. $R(aSb, cd) = acdb$.

We extend the domain of the adjunction operator S, defined in Section 2, so that in $S_\xi(X, Y)$, X, Y are strings over $A \cup \hat{A} \cup N \cup \hat{N} \cup \{S\}$.

DEFINITION 7.4. A *mixed adjunct language with null symbols* (MALN) is defined as $H(\hat{\Sigma}(\Sigma_e))$, where $\hat{\Sigma}$ is defined recursively as follows:

(i) If σ_i is in $\Sigma - \Sigma_S$, then $I(\sigma_i)$ is in $\hat{\Sigma}(\sigma_i)$. If σ_i is in Σ_S, then $I(\sigma_i)$ is in $\hat{\Sigma}_S(\sigma_i)$.

(ii)(a) If σ_i is in $\hat{\Sigma}_S(\sigma_j)$, σ_k is in $\hat{\Sigma}(\sigma_1)$ and $\langle \sigma_j, \sigma_1 \rangle$ is in R, then $R(\sigma_i, H(\sigma_k))$ is in $\hat{\Sigma}(\sigma_j)$.

(b) If σ_i is in $\hat{\Sigma}_S(\sigma_j)$, σ_k is in $\hat{\Sigma}(\sigma_1)$ and $(\sigma_j, \sigma_1, \xi)$ is in J, then $S_\xi(\sigma_i, H(\sigma_k))$ is in $\hat{\Sigma}_S(\sigma_j)$.

(c) If σ_i is in $\hat{\Sigma}(\sigma_j)$, σ_k is in $\hat{\Sigma}(\sigma_1)$ and $(\sigma_j, \sigma_1, \xi)$ is in J, then $S_\xi(\sigma_i, H(\sigma_k))$ is in $\hat{\Sigma}(\sigma_j)$.

(iii) Nothing else is in (σ_i), or $\hat{\Sigma}_S(\sigma_i)$ unless specified by (i) and (ii).

EXAMPLE 7.4. The language generated by the MAGN of Example 7.1 is

$$L = \{da^{n_1}cb^{n_1}a^{n_2}cb^{n_2} \cdots a^{n_k}cb^{n_k}d \mid k \geqslant 0;\ n_i \geqslant 1,\ 1 \leqslant i \leqslant k\}.$$

DEFINITION 7.5. A *simple replacement grammar with null symbols* (SRGN) is an MAGN where J, the set of adjunction rules, is empty. The language it generates is called a *simple replacement language with null symbols* (SRLN). (Note: Similarly an LAGN is an MAGN whose set of replacement rules is empty.)

THEOREM 7.2. *L is an* SRLN *iff L is a linear (ϵ-free) context-free language.*

The proof of Theorem 7.2 is straightforward and is given in Levy (1970). LALN's and SRLN's are incomparable, and both contain all regular sets. The language of arithmetic expressions, whose grammar was given earlier in this section, is an MALN but not an LALN or SRLN.

THEOREM 7.3. *There is a context-free language which can be generated by a mixed adjunct grammar but which cannot be generated by adjunction rules alone or replacement rules alone.*

Proof. The language

$$L = \{a^{n_1}pb^{n_1} \cdots a^{n_k}pb^{n_k}cd^{m_1}qe^{m_1} \cdots d^{m_l}qe^{m_l} \mid n_i \geqslant 2, m_i \geqslant 2, k \geqslant 1, l \geqslant 1$$

is generated by a mixed grammar with rules

$$\{(c, dSe, r_1), (c, aSb, 1_1), \langle aSb, aSb\rangle, \langle aSb, apb\rangle, \langle dSe, dSe\rangle, \langle dSe, dqe\rangle\}.$$

Local adjunction rules alone cannot generate this language by Theorem 7.1. Replacement rules alone cannot generate this language by Theorem 7.2. Q.E.D.

We state without proof the following.

THEOREM 7.4. *There is a context-free language which cannot be generated by combined use of replacement and local adjunction rules.*

The language $L = \{a^npb^ncd^mqe^m \mid m \geqslant 0, n \geqslant 0\}$ is such a language. In Levy (1970) an algorithm is given for forming a distributed adjunct grammar with null symbols (DAGN) for any mixed adjunct language. However, in Levy (1971), it was proved that, more generally, for any context-free language a DAGN can be effectively constructed, and the MALN's are a subclass of the class of context-free languages. In Joshi (1972c) the mixed adjunct grammars have been used as the base component of a transformational grammar.

8. CONCLUSION

The major results given here are Theorems 3.1 and 5.2, each of which resolves an open question in the theory of adjunct grammars. As a consequence of Theorem 3.1, which states that null symbols increase the generative power of local adjunct grammars, the relationship of the local adjunct grammars (without null symbols) to the regular sets is not known.

Section 4 clarifies the roles of left *and* right adjunction. In Section 6, the relationship of linear context-free and local adjunct languages is considered, and it is shown that the β-linear languages (Rosenberg, 1967) which are LALN's are all regular sets, although, in general, the class of linear context-free languages which are LALN's properly includes the class of regular sets (Hart (1973)).

Finally, in Section 7 the mixed adjunct grammars with local adjunction and simple replacement rules are defined and shown to generate a proper subset of the class of context-free languages.

APPENDIX

THEOREM 6.4. *Every β-linear* LALN *is regular.*

Proof. Every β-linear LALN, L, is expressible as a finite union $L = \bigcup_{i \text{ in } I} x_i \beta y_i$ where x_i, y_i are LALN's. L can then be rewritten as a union of disjoint terms, $L = \bigcup_{j,k} x_j \beta y_k$, where j, k range over all members of the power set of I, and $x_j \beta y_k$ denotes $(\bigcap_{i \text{ in } j} x_i) \beta (\bigcap_{i \text{ in } k} y_k)$. (e.g., $x_{\{1,2\}} \beta y_{\{2,3\}}$ denotes (strings in both x_1 and x_2) β (strings in both y_2 and y_3)). Note that $x_{j_1} \cap x_{j_2} = \varnothing$ if $j_1 \neq j_2$ and $y_{k_1} \cap y_{k_2} = \varnothing$ if $k_1 \neq k_2$. We claim that $x_p \beta y_q$ is contained in a regular set $R_{p,q}$, which is contained in L. Since there are only finitely many members of the power set of I, it will then follow that

$$L = \bigcup_{j,k} x_j \beta y_k \subseteq \bigcup_{j,k} R_{j,k} \subseteq L.$$

Hence, $L = \bigcup_{j,k} R_{j,k}$ is regular.

Next, we show that $x_p \beta y_q$ is contained in a regular set which is contained in L. Suppose that $x_p \beta y_q \subseteq l(y_q) \beta r(x_p) = L' \subseteq L$. Then L' is regular, since $l(y_q)$ and $r(x_p)$ are regular. Otherwise, $x_p \beta y_q \subseteq L$ (and, surely, $x_p \beta y_q \subseteq l(y_q) \beta r(x_p)$) and there are x_s, y_t such that $x_s \beta y_q \subseteq L$ and $x_p \beta y_t \subseteq L$ but $L_1 = l(y_q) \beta r(x_p)$ is not a subset of L. Hence, we must find a regular language, $R_{p,q}$, smaller than L_1, such that $R_{p,q}$ is contained in L. Let h_1 be the homomorphism on languages which maps everything in $l(y_t)$ into itself, and everything else to $\varnothing$, and h_2 be the homomorphism which maps everything in $r(x_s)$ into itself, and everything else to $\varnothing$. Now $x_p \beta y_q \subseteq h_1(l(y_q)) \beta h_2(r(x_p))$ and either $h_1(l(y_q)) \beta h_2(r(x_p)) \subseteq L$ or we may find a new pair of elements and repeat the construction.

Since 2^I is a finite set, the process must conclude after a finite number of steps. Q.E.D.

EXAMPLE 6.3. We illustrate the construction in the preceding theorem. Let $L = x_1 \beta y_1 \cup x_2 \beta y_2$ be a β-linear LALN, and let

$$x_1' = x_1 \cap \bar{x}_2 \qquad y_1' = y_1 \cap \bar{y}_2,$$
$$x_2' = \bar{x}_1 \cap x_2 \qquad y_2' = \bar{y}_1 \cap y_2,$$
$$x_{12}' = x_1 \cap x_2 \qquad y_{12}' = y_1 \cap y_2.$$

L may be represented by a modified Venn diagram, as shown in Fig. 5.

	y_1'	y_2'	y_{12}'
x_1'	$x_1'\beta y_1'$	////////	$x_1'\beta y_{12}'$
x_2'	////////	$x_2'\beta y_2'$	$x_2'\beta y_{12}'$
x_{12}'	$x_{12}'\beta y_{12}'$	$x_{12}'\beta y_2'$	$x_{12}'\beta y_{12}'$

FIG. 5. Diagrammatic representation of the language of Example 6.3.

The crosshatched areas of Fig. 5 are necessarily empty. (We assume, in this example, that all of the x_j', y_k' are nonempty. But the argument is essentially unchanged if some of these are empty.)

To show that $x_{12}'\beta y_{12}' \subseteq L' \subseteq L$ such that L' is regular: Note first that $1(y_{12}')\beta r(x_{12}')$ is not contained in L if all of the x_j', y_k' are nonempty. Let x_{12}', y_1' be the elements corresponding to x_s, y_t in the proof of Theorem 6.3. Then

$$r(x_{12}') = y_1' \cup y_2' \cup y_{12}',$$

$$1(y_1') = x_1' \cup x_{12}'.$$

Now let

$$h_1 : x_1' \mapsto x_1'; x_2' \mapsto \varnothing; x_{12}' \mapsto x_{12}',$$

$$h_2 : y_1' \mapsto y_1'; y_2' \mapsto y_2'; y_{12}' \mapsto y_{12}'.$$

Now $L_1 = x_1'\beta y_1' \cup x_1'\beta y_{12}' \cup x_{12}'\beta y_1' \cup x_{12}'\beta y_2' \cup x_{12}'\beta y_{12}'$. L_1 may be represented as shown in Fig. 6.

	y_1'	y_2'	y_{12}'
x_1'	$x_1'\beta y_1'$	////////	$x_1'\beta y_{12}'$
x_{12}'	$x_{12}'\beta y_1'$	$x_{12}'\beta y_{12}'$	$x_{12}'\beta y_{12}'$

FIG. 6. Diagrammatic representation of L_1 in Example 6.3.

Again $L' = h_1(l(y_{12}'))\,\beta\, h_2(r(x_{12}'))$ is not contained in L since $x_1'\beta y_2'$ is contained in L' but not in L. Choosing the elements x_{12}', y_2', we have

$$r(x_{12}') = y_1' \cup y_2' \cup y_{12}',$$

$$l(y_2') = x_{12}'.$$

Now applying the required homomorphism, we have

$$L_2 = x'_{12}\,\beta\,y_1' \cup x'_{12}\,\beta\,y_2' \cup x'_{12}\,\beta\,y'_{12} \text{ which is } \beta\text{-linear.}$$

Thus, x'_{12} is regular by Theorem 6.3, and $y_1' \cup y_2' \cup y'_{12}$ is regular by Theorem 6.3. Thus, $L_2 \subseteq L$ is regular. So $x'_{12}\,\beta\,y'_{12} \subseteq L_2 \subseteq L$, as claimed. Q.E.D.

Remarks. It can also be proved directly that if any β-linear language can be expressed as a finite union $L = \bigcup_i x_i\,\beta\,y_i$ (x_i, y_i languages) then L is regular. Note that this formulation does not mention LALN's and has Theorem 6.4 as a corollary.

RECEIVED: April 7, 1972

REFERENCES

HARRIS, Z. S. (1961), "String Analysis of Language Structure," Mouton, The Hague.

HARRIS, Z. S. (1968), "Mathematical Structure of Language," Mouton, The Hague.

HART, J. M. (1973), An infinite hierarchy of linear local adjunct languages, *J. Comput System Sci.*, to appear.

HWANG, R. (1972), private communication.

JOSHI, A. K. (1969), Properties of formal grammars with mixed types of rules and their linguistic relevance, *Proceedings of the Second International Symposium on Computational Linguistics*, Sanga Saby, Sweden.

JOSHI, A. K. (1972a), A class of transformational grammars, *in* "Formal Language Analysis" (M. Gross, M. Halle, and M. P. Schutzenberger, Eds.), Mouton (Series Janua Linguarum), The Hague.

JOSHI, A. K., KOSARAJU, S. R., AND YAMADA, H. (1972b), String adjunct grammars, *Information and Control* **21**, 93–116, 235–260.

JOSHI, A. K., LEVY, L. S., AND TAKAHASHI, M. (1972c), A tree generating system, *Proceedings IRIA Colloquium on the Theory of Automata, Languages, and Programming*, Paris.

JOSHI, A. K. AND TAKAHASHI, M. (1971), A characterization of the derivation trees of a context-free grammar and an intercalation theorem, *Moore School Technical Report*, University of Pennsylvania, Philadelphia, PA.

LEVY, L. S. (1970), Generalized local adjunction and replacement in adjunct languages, *Moore School Report No.* 70-29, University of Pennsylvania, Philadelphia, PA.

LEVY, L. S. (1971), Tree adjunct, parenthesis, and distributed adjunct grammars, *in* "Theory of Machines and Computations," pp. 127–143, Academic Press, New York.

ROSENBERG, A. L. (1967), A machine realization of the linear context-free languages, *Information and Control* **10**, 175–188.