On the Covering and Reduction Problems for Context-Free Grammars

JAMES N. GRAY AND MICHAEL A. HARRISON

University of California, Berkeley, California

ABSTRACT. A formal definition of one grammar "covering" another grammar is presented. It is argued that this definition has the property that G' covers G when and only when the ability to parse G' suffices for parsing G. It is shown that every grammar may be covered by a grammar in canonical two form. Every Λ-free grammar is covered by an operator normal form grammar while there exist grammars which cannot be covered by any grammar in Greibach form. Any grammar may be covered by an invertible grammar. Each Λ-free and chain reduced LR(k) (bounded right context) grammar is covered by a precedence detectable, LR(k) (bounded right context) reducible grammar.

KEY WORDS AND PHRASES: covers, reductions, parsing, precedence analysis, canonical precedence, context-free grammars

CR CATEGORIES: 4.12, 5.22, 5.23, 5.24

Introduction

There are parsing methods which require that the grammar under consideration be in some normal form or have some special property [3, 4, 5, 9, 13]. Sometimes, the requirement on the grammar is that it does not have a property such as not having left recursion for use with top-down parsing techniques. A few parsing methods are known which require no special form of the grammar (see, e.g., [2]), but this generality exacts a price in the complexity of the algorithm.

In the present paper, we consider a relationship between grammars called "covering." Intuitively, it will turn out that G' "covers" G when the ability to parse G' allows one to parse G by "table lookup techniques." The formal definitions will be more complicated than this because of some practical considerations, such as the desire to exclude productions which have no semantic significance. After justifying our definitions and comparing them with related concepts from the literature, we prove some positive results. We show that the canonical two form [1] can cover any grammar and that the operator normal form [4] can cover any Λ-free grammar. It is also shown that any grammar may be covered by an invertible grammar. A typical negative result is that there are grammars which cannot be covered by any grammar in Greibach form [10].

This research was supported in part by NSF Grant GJ 474.

Authors' present addresses: J. N. Gray, IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598; M. A. Harrison, Department of Computer Science, University of California, Berkeley, CA 94720.

Attention is then turned to the class of bottom-up parsing methods. Bottom-up parsing may be regarded as the iteration of a two-step process: detecting a phrase and then reducing it. It is shown that each step may be trivialized at the expense of the other. It is shown that every LR(k) grammar [13] may be covered by a grammar which is precedence detectable and LR(k) reducible. A similar result holds when "LR(k)" is replaced by "bounded right context" [5].

The present paper is organized as follows: the remainder of this Introduction contains most of our formal definitions and notational conventions. Section 1 introduces covers and discusses their connection with other relations between grammars. We then show that each grammar is covered by a canonical two form grammar. It is proven that each Λ-free grammar is covered by a grammar in operator form. This statement becomes false if the Λ-free hypothesis is omitted. It is shown that there is a grammar which cannot be covered by any grammar in Greibach normal form. Invertibility is introduced and it is shown that every grammar can be covered by an invertible grammar by abusing Λ-rules. The ramifications of this are explored.

In Section 2, bottom-up parsing is dichotomized and it is shown that each Λ-free and chain reduced LR(k) (bounded right context) grammar is covered by a precedence detectible, LR(k) (bounded right context) reducible grammar.

Section 3 concludes and summarizes the discussion.

We now begin to list some of the formal definitions which are required.

Definition. A *context-free grammar* is a 4-tuple $G = (V, \Sigma, P, S)$ where:

(i) V is a finite nonempty set (*vocabulary*),
(ii) $\Sigma \subseteq V$ is a finite nonempty set (*terminal symbols*),
(iii) $N = V - \Sigma$ is the set of *variables* and $S \in N$,
(iv) P is a finite subset[1] of $N \times V^*$ and we write $u \to v$ in P instead of $(u,v) \in P$.

P is the set of *productions.*

It is convenient to introduce a general notation concerning relations.

Definition. Let ρ be a binary relation on a set X, i.e. $\rho \subseteq X \times X$. Define $\rho^0 = \{(a,a) \mid a \in X\}$, and for each[2] $i \geq 0$, $\rho^{i+1} = \rho^i\rho$. Lastly, $\rho^* = \bigcup_{i\geq 0} \rho^i$ and $\rho^+ = \rho^*\rho$. For a binary relation ρ on X, ρ^* is the *reflexive-transitive closure* of ρ while ρ^+ is the *transitive closure* of ρ.

Next, we can define the rules for rewriting strings.

Definition. Let $G = (V, \Sigma, P, S)$ be a context-free grammar and let $u, v \in V^*$. Define $u \Rightarrow v$ if there exist words $x, y, w \in V^*$ and $A \in N$ so that $u = xAy$, $v = xwy$, and $A \to w$ is in P. If $y \in \Sigma^*$, we write $u \underset{R}{\Rightarrow} v$. Furthermore, define

$$\overset{*}{\Rightarrow} = (\Rightarrow)^* \quad \text{and} \quad \overset{*}{\underset{R}{\Rightarrow}} = (\underset{R}{\Rightarrow})^*.$$

A string $x \in V^*$ is said to be a *sentential form* if $S \overset{*}{\Rightarrow} x$ and a *canonical sentential form* if $S \overset{*}{\underset{R}{\Rightarrow}} x$. Not every sentential form is canonical.

The set $L(G) = \{x \in \Sigma^* \mid S \overset{*}{\Rightarrow} x\}$ is the *language generated by* G.

We now mention four similar but notationally different definitions of derivations.

[1] Let X and Y be sets of words. Write $XY = \{xy \mid x \in X, y \in Y\}$ where xy is the concatenation of x and y. Define $X^0 = \{\Lambda\}$ where Λ is the null word. For each $i \geq 0$, define $X^{i+1} = X^iX$ and $X^* = \bigcup_{i\geq 0} X^i$. Let $X^+ = X^*X$ and let $\varnothing$ denote the empty set. Finally, if x is a string, let $\lg(x)$ denote the length of x which is the number of occurrences of symbols in x.

[2] The operation is a *composition* of relations which is defined as follows: if $\rho \subseteq X \times Y$ and $\sigma \subseteq Y \times Z$, define $\rho\sigma = \{(x,z) \mid (x,y) \in \rho \text{ and } (y,z) \in \sigma \text{ for some } y \in Y\}$. Observe that $\rho\sigma \subseteq X \times Z$.

If $u_0 \Rightarrow u_1 \Rightarrow \cdots \Rightarrow u_r$ then we say that the sequence $(u_0, \cdots, u_r)$ is a *derivation* of u_r *from* u_0. If $u_0 \underset{R}{\Rightarrow} u_1 \underset{R}{\Rightarrow} \cdots \underset{R}{\Rightarrow} u_r$ the derivation is said to be a *canonical derivation.* If for each $0 \leq i < r$, if $u_i = v_i A_i w_i$ and $u_{i+1} = v_i y_i w_i$ and u_{i+1} may be obtained from u_i by using production $\pi_i = A_i \rightarrow y_i$ at position $n_i = \lg(v_i y_i)$, we say that the sequence of rule-integer pairs $((\pi_0, n_0), \cdots, (\pi_{r-1}, n_{r-1}))$ is a *derivation* of u_r *from* u_0. In the case where this derivation is canonical the n_i are superfluous, so we also let $(\pi_0, \cdots, \pi_{r-1})$ denote the canonical derivation of u_r from u_0. If u_0 is not mentioned, it is assumed that $u_0 = S$. Any particular derivation also corresponds to a labeled directed tree, called the *parse tree.*

If the sequence $(u_0, \cdots, u_r)$ is a derivation of u_r from u_0 then $(u_r, \cdots, u_0)$ is said to be a *parse* of u_r to u_0. If the derivation is canonical then the parse is said to be *canonical.* If u_0 is not mentioned then we assume that $u_0 = S$.

If $(s_1, \cdots, s_n)$ is any sequence, it may be denoted by $(s_i)_{i=1}^n$. If P is some predicate defined on the s_i then the subsequence of those s_i satisfying P is denoted by $(s_i \mid P(s_i))_{i=1}^n$. If f is a function on the s_i then the sequence $(f(s_1), \cdots, f(s_n))$ is denoted by $(f(s_i))_{i=1}^n$.

In a particular derivation of a canonical sentential form x, denoted by a sequence $((\pi_0, n_0), \cdots, (\pi_r, n_r))$, if $\pi_r = (A \rightarrow y)$ then the occurrence of the substring y in x at position n_r is a *simple phrase* of x, and the pair (π_r, n_r) is called a *reduction* of x. If the derivation is canonical then (π_r, n_r) is called a *handle* of x.

Let Σ and Δ be two alphabets and suppose f is a function from Σ into Δ^*. f may be extended (uniquely) to a monoid *homomorphism* from Σ^* into Δ^* by the conditions

$$f(\Lambda) = \Lambda, \qquad f(a_1, \cdots, a_n) = f(a_1), \cdots, f(a_n)$$

for $a_i \in \Sigma$ for $1 \leq i \leq n$. If $L \subseteq \Sigma^*$, define $f(L) = \{f(x) \mid x \in L\}$. If L is context-free (regular) and f is a homomorphism, then $f(L)$ is context-free (regular) [6, 11].

We will be considering a number of special properties of grammars and we now list some of these. Many of these definitions are in standard textbooks on language theory [6, 11].

Definition. A context-free grammar $G = (V, \Sigma, P, S)$ is said to be

(i) Λ-*free* if $P \subseteq N \times V^+$,

(ii) *chain-free*[3] if $P \cap (N \times N) = \varnothing$,

(iii) *reduced* if

(a) for each $A \in V$, there exist $x, y \in V^*$ so that $S \overset{*}{\Rightarrow} xAy$, and

(b) for each $A \neq S$ there exists $x \in \Sigma^*$ so that $A \overset{*}{\Rightarrow} x$,

(iv) in *operator form* if $P \subseteq N \times (V^* - V^*N^2V^*)$,

(v) in *canonical two form* if $P \subseteq N \times (\{\Lambda\} \cup V \cup N^2)$,

(vi) in *Greibach form* if $P \subseteq N \times \Sigma V^*$.

The following results are well known:

(a) Every context-free language not containing Λ has a Λ-free grammar.

(b) Every context-free language has a context-free grammar which is chain-free.

(c) Every context-free language has a reduced context-free grammar.

(d) Every context-free language has a grammar in operator form [10].

(e) Every context-free language has a grammar in canonical two form [1].

(f) Every context-free language not containing Λ has a context-free grammar in Greibach form [10].

[3] A derivation $Z_0 \Rightarrow \cdots \Rightarrow Z_r$ is said to be a *chain* if $r > 0$ and $Z_i \in N$ for $0 \leq i \leq r$.

These results may be combined into pairs (i.e. a grammar may be assumed to satisfy an arbitrary pair of the properties) except that pairs (d,e) and (e,f) are incompatible.

1. *Basic Results*

In the present section we consider a number of alternate definitions of "covering" and other relations between grammars. We arrive at a definition which turns out to be quite useful and captures the intuitive notion of "covering" with respect to parsing. That is, if G' covers G and if one can parse G', then one can parse G.

Our first definition is a familiar and weak concept from language theory.

Definition. Two context-free grammars G and G' are said to be *equivalent* if $L(G) = L(G')$.

For our remaining definitions, we need the following framework. Let $G = (V, \Sigma, P, S)$ and $G' = (V', \Sigma, P', S')$ be two context-free grammars over Σ. Let f be any map from V' into V which is the identity on Σ, i.e. $f(a) = a$ for each $a \in \Sigma$. Extend f to be a (monoid) homomorphism from $(V')^*$ into V by requiring $f(xy) = f(x)f(y)$ for each $x, y \in (V')^*$.

Notation. For any set P' of productions, write

$$f(P') = \{f(A) \to f(x) \mid A \to x \text{ is in } P'\}.$$

The following definition offers another relationship between grammars.

Definition. Let G, G', and f be as above. We say that f is a *homomorphism* from G' onto G if

(a) $f(S') = S$,

(b) $f(P') = P$.

If f is also one-to-one then f is an *isomorphism*.

The original notion of "covering" was due to John Reynolds (cf. [15]). We shall also introduce "weak covers."

Definition. Let G, G', and f be as above. G is said to be a *weak Reynolds cover* of G' under f if

(a) $f(S') = S$, and

(b) $f(A) \overset{*}{\Rightarrow} f(x)$ in G if $A \to x$ is in P'.

Finally, we consider a strengthened version of the previous definition of covering which is the original one [15].

Definition. Let G, G', and f be as above. G is said to *Reynolds cover* G' under f if

(a) $f(S') = S$, and

(b) $f(P') \subseteq P$.

These definitions have all been used in the literature [7, 15]. Some of the simple formal relations among the definitions are as follows:

PROPOSITION. *Let G, G', and f be as before.*

(*a*) *If f is an isomorphism of G' onto G then f is a homomorphism of G' onto G.*

(*b*) *If f is a homomorphism from G' onto G then G is a Reynolds cover of G' under f.*

(*c*) *If G is a Reynolds cover of G' under f then G is a weak Reynolds cover of G' under f.*

(*d*) *If f is a homomorphism of G' onto G then G is equivalent to G'.*

None of these definitions seems to capture the notion that we think is essential for programming applications. We would like to say G' covers G if given a parser for G' one can construct a parser for G. The motivation for this is that parsers typically handle grammars in some normal form. Presented with an arbitrary grammar G it may be possible to transform it into a grammar G' which is in this normal form. In what cases can a parser for G' be used to produce a parser for G?

For example, simple top-down parsers will not tolerate left recursive rules which allow $A \overset{+}{\Rightarrow} Ax$ for some nonterminal A and string x. However, given a grammar G there is a grammar G' equivalent to G which has no such left recursive rules. Can one construct a parser for G given a parser for G'? We shall prove that the answer is no, given our definition of covering.

The notion of equivalence of grammars is too weak since it makes no reference at all to the parses in G' and G. For example G may be ambiguous and G' not. On the other hand, the notion of isomorphism is too strong. It means that parses in G and G' differ only by renaming of nonterminals. G' is almost identical to G in this case. The definitions of covers which are due to Reynolds come considerably closer to an acceptable definition of "similar." Reynolds shares our desire to characterize the process of transforming a grammar G into a grammar G' which is easier to parse than G and which has parses similar to those in G. However Reynolds' emphasis is significantly different. He does not even require $L(G) = L(G')$; he merely requires $L(G) \subseteq L(G')$. Reynolds intends to have the semantic routines detect and reject those strings in $L(G') - L(G)$. For example, every canonical two form grammar G over terminal alphabet $\{0, 1\}$ is covered in Reynolds' sense by the grammar

$$S \to SS \mid S \mid 0 \mid 1 \mid \Lambda$$

Thus the semantic routines will do *all* the work in this case. Although the canonical two form of G is weakly Reynolds covered by G, in general no Greibach normal form or operator normal form of a Λ-free version of G is (weakly) Reynolds covered by G.

Before presenting our notion of covering, we must generalize the idea of generation because of the following practical considerations. In most[4] formal treatments of parsing, the parser must enumerate *all* the nodes of the parse tree. In programming practice, certain nodes of the parse tree have no semantic significance and do not need to be present in a similar grammar. For example, consider the generation tree of Figure 1 which occurs in EULER [16].

The chain **expr-** $\overset{*}{\Rightarrow} \lambda$ is typical of what happens in grammars for programming languages. Chains exist to enforce precedence among operators and to collect several categories of syntactic types (e.g. in ALGOL ⟨statement⟩ → ⟨unconditional statement⟩ | ⟨conditional statement⟩ | ⟨for statement⟩).

Chain productions rarely have semantic significance. In our running example, only the following productions have nontrivial semantics:

$$\textbf{expr-} \to \textbf{var} \leftarrow \textbf{expr-}$$

$$\textbf{var-} \to \lambda$$

$$\textbf{primary} \to \textbf{var}$$

$$\lambda \to A$$

$$\lambda \to B$$

For the purposes of code generation, the "sparse generation tree" of Figure 2(a) is as satisfactory as the tree in Figure 1. The tree shown in Figure 2(b) would not be a

[4] Floyd precedence [4] is a parsing scheme which makes this explicit. Only rules containing terminal characters are enumerated.

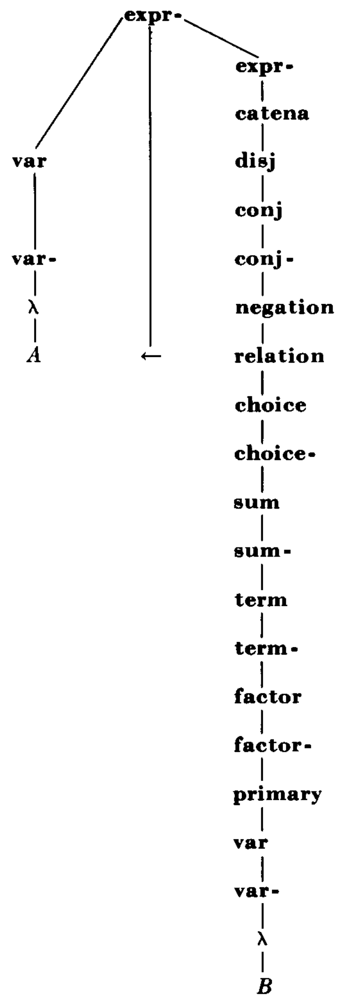


FIG. 1. A generation from EULER.

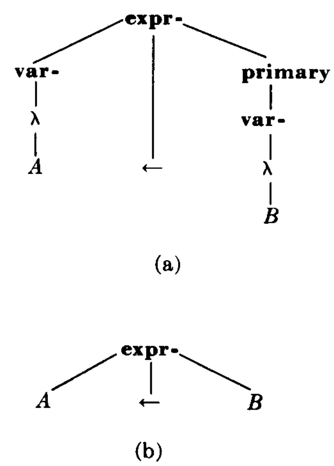


FIG. 2. (a) A sparse parse of the generation in Figure 1; (b) a nonsparse parse of Figure 1.

satisfactory replacement for the original tree because some nodes with semantic significance have been omitted.

Further, nodes of G' may be superfluous because G' has more structure than G. For example, many rules of the canonical two form of a grammar exist only to produce a binary parse tree.

We can formalize these notions by assuming that, independent of context, a production either does or does not have semantic significance. If it does not, it may be omitted from the parse. In what follows, think of H as the set of those productions of G with semantic significance and $P - H$ as those productions with no semantic significance.

Definition. Let $G = (V, \Sigma, P, S)$ be a grammar and let $H \subseteq P$. Let

$$D = (A_i \to x_i)_{i=1}^{n}$$

be a canonical derivation in G. Then the corresponding *H-sparse derivation* is

$$D_H = (A_i \to x_i \mid A_i \to x_i \text{ is in } H)_{i=1}^{n}.$$

Let $CD(G, H)$ denote the set of all such H-sparse derivations in G.

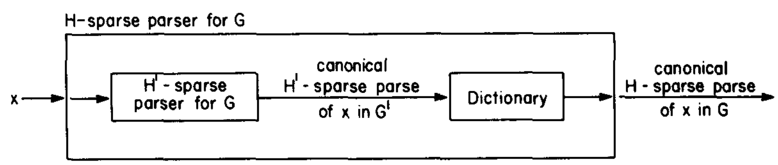
Note that if $H = \varnothing$, D_H is the null sequence, and that in general D_H is not a derivation. It is simply the subsequence of steps of D involving productions of H. As usual, by inverting the index i, one obtains parses from derivations. In particular $((A_i \to x_i, n_i) \mid A_i \to x_i \in H)_{i=n}^{1}$ will be called the corresponding *H-sparse parse of x*.

For H as described above, the H-sparse generation of Figure 1 is

$$((\mathbf{expr\text{-}} \to \mathbf{var} \leftarrow \mathbf{expr\text{-}},3),$$
$$(\mathbf{primary} \to \mathbf{var\text{-}},3),$$
$$(\mathbf{var\text{-}} \to \lambda,3),$$
$$(\lambda \to B,3),$$
$$(\mathbf{var\text{-}} \to \lambda,1),$$
$$(\lambda \to A,1)).$$

We reformulate the parsing problem as follows: given a grammar G and a set $H \subseteq P$, produce a parser which, for each $x \in \Sigma^*$, enumerates all canonical H-sparse parses with respect to G.

In this light, parsing G' will be as good as parsing G if for some $H' \subseteq P'$ one can easily construct all canonical H-sparses in H for x from all canonical H'-sparse parses of x in G'. Schematically



We are finally ready to present our definition of cover.

Definition. Let $G = (V, \Sigma, P, S)$ and $G' = (V', \Sigma, P', S')$ be context-free grammars. Let $H \subseteq P$ and $H' \subseteq P'$. Let φ be a map from H' into H. For any canonical derivation $D = (A_i \to x_i)_{i=1}^{n}$ in G' of some $x \in \Sigma^*$, define the *image of D under φ* to be $\varphi(D) = (\varphi(A_i \to x_i) \mid A_i \to x_i \text{ is in } H')_{i=1}^{n}$. $\varphi(D)$ is an element of H^*. (G', H') is said to *cover* (G, H) *under* φ iff

(a) $L(G) = L(G')$, and

(b) for each $x \in L(G)$,

(i) if D is an H-sparse derivation of x in G then there is an H'-sparse derivation D' of x in G' so that $\varphi D' = D$, and

(ii) if D' is an H'-sparse generation of x in G' then φD is an H-sparse generation of x in G.

G' is said to *cover* (G, H) if some H' and φ exist such that (G', H') covers (G, H) under φ. If G' covers (G, P) we say G' *completely covers* G.

We remark immediately that these relations are reflexive and transitive but not symmetric in general; thus they are not equivalence relations. Note that our definition of cover runs in the other direction to Reynolds, i.e. we say G' covers G while Reynolds says that G covers G'. We now summarize some of the simple properties of covers.

PROPOSITION. *(a) G' covers $(G, \varnothing)$ if and only if G and G' are equivalent.*

(b) If (G', P') covers (G, P) then the degree[5] *of ambiguity in G' and G on any string $x \in \Sigma^*$ is the same.*

(c) If G' is isomorphic to G (under φ) then G' covers G (under φ).

(d) If (G'', H'') covers (G', H') under φ' and (G', H') covers (G, H) under φ, then (G'', H'') covers (G, H) under $\varphi\varphi'$.

Thus covers provide a spectrum of relationships as H and H' vary.

It should be noted that we can find grammars G' and G and a map f so that f is a homomorphism of G onto G' and G' does not cover G (in fact $L(G') \neq L(G)$). For example, choose G' to have the productions:

$$S' \to Ta|a$$

$$T \to b$$

G has the productions

$$S \to Sa|a|b$$

and φ is the function which takes S' and T onto S and is the identity on $\{a, b\}$. Clearly $L(G') \neq L(G)$.

One might think that if G covers G' and if G' covers G then G and G' are very similar. Consider the following two grammars.

$$\begin{array}{cc} G: & G': \\ S \to Ab & S \to aB \\ A \to a & B \to b \end{array}$$

Clearly G covers G' and G' covers G yet G and G' are not isomorphic. Indeed the trees are quite different. Many other examples of this type exist and when null rules are used, the trees may differ radically.

Before using covers in a treatment of bottom-up parsing we first explore the relationship between grammars and their more common normal forms.

THEOREM 1.1. *Each context-free grammar G is completely covered by a grammar G' which is in canonical two form.*

PROOF. Let [and] be two new symbols not in the vocabulary of $G = (V, \Sigma, P, S)$. Define $G' = (V', \Sigma, P', S')$ by:

$N' = \{[y] \mid A \to xy \text{ is in } P \text{ for some } x \in V^*, y \in V^2V^*\} \cup \{[A] \mid A \in V\}$,
$P_1 = \{[A] \to \Lambda \mid A \to \Lambda \text{ is in } P\}$,
$P_2 = \{[A] \to [B] \mid A \to B \text{ is in } P, \text{ for some } A \in N, B \in V\}$,
$P_3 = \{[A] \to [B][x] \mid A \to Bx \text{ is in } P \text{ for some } A, B \in V, x \in V^+\}$,
$P_4 = \{[Ax] \to [A][x] \mid A \in V \text{ and } x \in V^+ \text{ and } [Ax] \in N'\}$,
$P_5 = \{[a] \to a \mid a \in \Sigma\}$,
$P' = P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5$,
$S' = [S]$.

[5] Let $G = (V, \Sigma, P, S)$ be a grammar and $x \in \Sigma^*$. The *degree of ambiguity* of x is the number of canonical derivations of x in G.

Let $H' = P_1 \cup P_2 \cup P_3$ and define φ by cases:

$\varphi([A] \to \Lambda) = A \to \Lambda$ for each $[A] \to \Lambda$ in P_1,
$\varphi([A] \to [B]) = A \to B$ for each $[A] \to [B]$ in P_2,
$\varphi([A] \to [B][x]) = A \to Bx$ for each $[A] \to [B][x]$ in P_3.

Clearly $G' = (V', \Sigma, P', S')$ is a grammar in canonical two form and φ is a bijection from H' onto P.

CLAIM 1. *Let* $(A_i \to x_i)_{i=1}^n$ *be a canonical derivation,* $[A] \overset{*}{\underset{R}{\Rightarrow}} x$ *in* G' *where* $x \in \Sigma^*$ *and* $A \in N$. *Then its image under* φ, $(\varphi(A_i \to x_i) \mid A_i \to x_i \text{ is in } H')_{i=1}^n$, *is a canonical derivation* $A \overset{*}{\underset{R}{\Rightarrow}} x$ *in* G.

PROOF. Consider the context-sensitive grammar obtained by deleting the brackets [and] from the productions of G'. P_4 and P_5 now yield identity transformations and productions in H' yield the transformations of P. Using this fact it follows that the image under φ of each canonical derivation in G' is a canonical derivation in G since φ simply discards the brackets on productions in H'.

The converse is less straightforward. First we note:

CLAIM 2. (*a*) *If* $A \to \Lambda$ *is in* P *then* $[A] \to \Lambda$ *is in* P'.

(*b*) *If* $A \to a$ *is in* P *then* $[A] \to [a]$ *is in* P'.

(*c*) *If* $A \to B$ *is in* P *then* $[A] \to [B]$ *is in* P'.

(*d*) *If* $A \to B_1 \cdots B_n$ *is in* P *for* $n > 1$ *where* $A, B_1, \cdots, B_n \in V$ *then* $[A] \underset{R}{\Rightarrow} [B_1] \cdots [B_n]$ *in* G' *by a derivation* $(C_i \to x_i)_{i=1}^{n-1}$, *where* $C_1 = [A]$ *and* $C_i = [B_i \cdots B_n]$ *for* $1 < i < n$ *and* $x_i = [B_i][B_{i+1} \cdots B_n]$ *for* $1 \le i < n$, *and in each case the image under* φ *of the derivation in* G' *is the derivation in* G.

PROOF. Each case may be verified by inspection of P'. The conclusion is immediate in cases (a), (b), and (c). In case (d) the image of $(C_i \to x_i)_{i=1}^{n-1}$ is $(A \to B_1 \cdots B_n)$ since $C_1 \to x_1$ is in P_3 and $C_i \to x_i$ is in P_4 for $1 < i < n$.

CLAIM 3. *Let* $A_1, \cdots, A_m \in V$ *and* $x \in \Sigma^*$. *If* $A_1 \cdots A_m \overset{*}{\underset{R}{\Rightarrow}} x$ *in* G *by derivation* $D = (C_i \to x_i)_{i=1}^n$ *then* $[A_1][A_2] \cdots [A_m] \overset{*}{\underset{R}{\Rightarrow}} x$ *in* G' *by a derivation*[6]

$$D' = (C_{ij} \to x_{ij})_{j=1}^{m_i}{}_{i=1}^{n}$$

such that the image of D' *under* φ *is* D.

PROOF. The argument is an induction on n.

Basis. $n = 0$. In this case $A_1 \cdots A_m \overset{*}{\underset{R}{\Rightarrow}} x$ in G by $D = \Lambda$. Thus $A_1 \cdots A_m = x \in \Sigma^*$. By using rules in P_4, we have

$$[A_1] \cdots [A_m] \underset{R}{\Rightarrow} [A_1] \cdots [A_{m-1}]A_m \overset{*}{\underset{R}{\Rightarrow}} A_1 \cdots A_m$$

in G' by the derivation $D' = ([A_{m-i+1}] \to A_{m-i+1})_{i=1}^m$. But $\varphi(D') = \Lambda$ and the basis is established.

Induction Step. Suppose Claim 3 holds for $i < n$ and consider the case $i = n$. In particular, suppose $A_1 \cdots A_m \underset{R}{\Rightarrow} B_1 \cdots B_p$ in G by $(C_1 \to x_1)$. Then by Claim 2, we know that there exists a canonical derivation $(C_{ij} \to x_{ij})_{j=1}^{m_1}$ for $[A_1][A_2] \cdots [A_m] \overset{*}{\underset{R}{\Rightarrow}} [B_1] \cdots [B_p]$ in G' and that its image under φ is $(C_1 \to x_1)$. By hypothesis, there is a canonical derivation $(C_{ij} \to x_{ij})_{j=1}^{m_i}{}_{i=2}^{n}$ for $[B_1] \cdots [B_p] \overset{*}{\underset{R}{\Rightarrow}} x$ in G' which has $(C_i \to x_i)_{i=2}^n$ as its image under φ. Thus $(C_{ij} \to x_{ij})_{j=1}^{m_i}{}_{i=1}^{n}$ satisfies Claim 3 and the induction is complete.

[6] If $n = 0$ then $D' = (C_{1j} \to x_{1j})_{j=1}^{m_1}$ by convention.

Claim 1 shows that every canonical derivation in G' of $x \in \Sigma^*$ from $[S]$ has a canonical derivation of x from S in G as its image under φ. Conversely, Claim 3 shows that this map is onto all derivations in G. This shows that $L(G') = L(G)$ and so G' covers G under φ. ▮

Theorem 1.1 demonstrates the necessity for the concept of sparse derivations. Although G' and G are very similar, G' is not isomorphic to G nor does G (weakly) Reynolds cover G'. However G does weakly Reynolds cover G'. The following result differentiates between covers and weak Reynolds covers. The Λ-free version, G' of a grammar G covers G up to Λ-rules [i.e. $P - H = \{A \to \Lambda \text{ in } P\}$] provided that G and G' have the same degree of ambiguity; on the other hand G does not weakly Reynolds cover G'.

Another commonly encountered normal form is the operator normal form grammar. It plays an important role in precedence analysis [3, 4, 9]. Greibach [10] originally showed that every grammar could be transformed to an equivalent grammar in operator normal form. However it is known that this transformation drastically changes the structure of the parse tree. It was conjectured that the reason that Floyd's precedence scheme is weaker than the scheme of Wirth and Weber was that it was impossible to get covering grammars that are in operator normal form. This conjecture proved to be false as the next result shows. One should consult [9] for a further discussion of this point.

THEOREM 1.2. *Every Λ-free grammar is completely covered by a grammar in operator normal form.*

PROOF. Let $G = (V, \Sigma, P, S)$ be a context-free grammar. We may assume, without loss of generality, that G is in canonical two form by using Theorem 1.1 and the transitivity of covers.

Let $G = (V', \Sigma, P', S)$ where $V' = \{S\} \cup \Sigma \cup (N \times \Sigma)$ and define $P' = P_1 \cup P_2 \cup P_3 \cup P_4$ as follows:

$$P_1 = \{S \to (S,a)a \mid a \in \Sigma\},$$
$$P_2 = \{(A,a) \to \Lambda \mid A \in N,\ a \in \Sigma,\ A \to a \text{ in } P\},$$
$$P_3 = \{(A,a) \to (B,a) \mid A, B \in N;\ a \in \Sigma,\ A \to B \text{ in } P\},$$
$$P_4 = \{(A,a) \to (B,b)b(C,a) \mid A, B, C \in N;\ a, b \in \Sigma;\ A \to BC \text{ in } P\}.$$

Next we define $H' = P_2 \cup P_3 \cup P_4$, and φ is defined by cases.

$$\varphi((A,a) \to \Lambda) = A \to a \text{ if } (A,a) \to \Lambda \text{ is in } P_2,$$
$$\varphi((A,a) \to (B,a)) = A \to B \text{ if } (A,a) \to (B,a) \text{ is in } P_3,$$
$$\varphi((A,a) \to (B,b)b(C,a)) = A \to BC \text{ if } (A,a) \to (B,b)b(C,a) \text{ is in } P_4.$$

We must show that (G', H') covers (G, P) under φ. To do this, we establish a claim.

CLAIM. *For each $a \in \Sigma$, $x \in \Sigma^*$, $A \in N$, $(A,a) \overset{*}{\Rightarrow} x$ in G' by a canonical derivation $(\pi_i')_{i=1}^n$ if and only if $A \overset{*}{\Rightarrow} xa$ in G by canonical derivation $\varphi((\pi_i')_{i=1}^n)$.*

PROOF. The argument is an induction on n.

Basis. If $n = 1$, then $x = \Lambda$ and $(A,a) \to \Lambda$ is in P'. This holds if and only if $A \to a$ is in P which completes the basis.

Induction Step. Assume the result for $1 \leq n < k$ and consider the case $n = k$. Since $n = k > 1$, $\pi_1' \in P_3 \cup P_4$. There are two cases depending on whether $\pi_1' \in P_3$ or $\pi_1' \in P_4$. We will give the details only in case $\pi_1' \in P_4$ and leave the (easier) case of $\pi_1' \in P_3$ to the reader. If $\pi_1' = (A,a) \to (B,b)b(C,a)$ then $\varphi(\pi_1') = A \to BC$

by construction. There is some j so that $(\pi_i')_{i=2}^{j}$ is a canonical derivation of $(C,a) \underset{R}{\overset{G'}{\Rightarrow}}{}^* x_2$ and $(\pi_i')_{i=j+1}^{n}$ is a canonical derivation of $(B,b) \underset{R}{\overset{G'}{\Rightarrow}}{}^* x_1$ where $x = x_1bx_2$. By the induction hypothesis, these canonical derivations exist if and only if $\varphi((\pi_i')_{i=2}^{j})$ is a canonical derivation of $C \underset{R}{\overset{G}{\Rightarrow}}{}^* x_2a$ and $\varphi((\pi_i')_{i=j+1}^{n})$ is a canonical derivation of $B \underset{R}{\overset{G}{\Rightarrow}}{}^* x_1b$. Combining these results,

$$(A,a) \overset{G'}{\Rightarrow} (B,b)b(C,a) \underset{R}{\overset{G'}{\Rightarrow}}{}^* x_1bx_2$$

if and only if

$$A \overset{G}{\Rightarrow} BC \underset{R}{\overset{G}{\Rightarrow}}{}^* x_1bx_2a.$$

This extends the induction and completes the proof of the claim.

From the claim it follows that for any $x \in \Sigma^*$; $a \in \Sigma$;

$$S \Rightarrow (S,a)a \overset{*}{\Rightarrow} xa \text{ in } G'$$

by derivation $(\pi_i')_{i=1}^{n}$ if and only if $S \overset{*}{\Rightarrow} xa$ in G' by canonical derivation $\varphi(\pi_i')_{i=1}^{n}$. This shows that $L(G') = L(G)$ and that φ is a map from $CD(G', H')$ onto $CD(G, P)$. Therefore G' covers G under φ. ∎

We note that a slightly more complex construction for G' would yield a Λ-free grammar. The strongest result we can state is that every Λ-free grammar is completely covered by a Λ-free operator grammar. Furthermore, this construction preserves Floyd precedence relations [4, 9].

The statement of the previous theorem immediately suggests the question of whether the hypothesis of Λ-freeness can be dropped. We will now show that it cannot be omitted and this will be our first real result of a negative character.

THEOREM 1.3. *There is a context-free grammar G which is not covered by any operator normal form grammar.*

PROOF. Let G be the grammar whose rules are:

$$S \rightarrow SS | \Lambda$$

Suppose that $G' = (V', \Sigma, P', S')$ is an operator normal form grammar which covers G under φ. There is no loss of generality in assuming that G' is reduced.

Let $CD(G, P)$ denote the set of canonical derivations of Λ in G and let $CD(G', P')$ be the set of canonical derivations of Λ in G'.

CLAIM 1. *$CD(G', P')$ is a regular set.*

PROOF. Suppose $A \rightarrow x$ is in P'. Since G' is reduced and since $L(G') = L(G) = \{\Lambda\}$, x cannot contain any characters of Σ. Because G' is in operator normal form, x cannot contain two adjacent nonterminals, so $x \in \{\Lambda\} \cup \{N'\}$. Thus P' consists entirely of chain rules and Λ-rules, i.e. $P' \subseteq (N') \times (\{\Lambda\} \cup (N'))$. It follows easily by induction that

$$CD(G', P') = \{ (A_i \rightarrow x_i)_{i=1}^{n} | n \geq 1; A_0 = S; x_n = \Lambda; A_i \rightarrow x_i \text{ is in } P', \text{ and } x_i = A_{i+1} \text{ for } 1 \leq i < n\}.$$

Let

$$R_1 = (\{S \rightarrow x \text{ in } P'\} (P')^* \{A_i \rightarrow \Lambda \text{ in } P'\}) \cup \{S' \rightarrow \Lambda | S' \rightarrow \Lambda \text{ is in } P'\}.^{7}$$

Define

$$R_2 = P' \cup \{ (A_i \rightarrow x_i)_{i=1}^{n} \in (P')^* | n \geq 1, x_i = A_{i+} \text{ for } 1 \leq i < n\}.$$

[7] That is, $S' \rightarrow \Lambda$ is in R_1 if and only if it is in P'.

R_1 is clearly a regular set and R_2 is also regular since it is a special type of regular set. (It is essentially the set of all R-sequences of a finite set R, i.e. $\{a_1 \cdots a_n \mid n \geq 1, (a_i, a_{i+1}) \in R, 1 \leq i < n\}$. Such sets are well known to be regular [6, 11].) But $CD(G', P') = R_1 \cap R_2$, and so $CD(G', P')$ is regular since R_1 and R_2 are.

CLAIM 2. *$CD(G, P)$ is not regular.*

PROOF. By a straightforward induction one can verify that

$$CD(G, P) = \{x \in \{S \to SS, S \to \Lambda\}^* \mid \text{for every}^8\ k,\ 1 \leq k \leq \lg(x),\ \#_{S\to SS}({}^{(k)}x) \geq \#_{S\to\Lambda}({}^{(k)}x);\ \#_{S\to SS}(x) + 1 = \#_{S\to\Lambda}(x)\}.$$

Suppose that $CD(G, P)$ were regular. Then if we let $R = \{S \to SS\}^*\{S \to \Lambda\}^*$, then $CD(G, P) \cap R$ would be regular. But

$$CD(G, P) \cap R = \{(S \to SS)^i(S \to \Lambda)^{i+1} \mid i \geq 1\},$$

which is clearly not regular. This contradicts the supposition that $CD(G, P)$ is regular.

To complete the proof, assume that G' covers G under φ. Note that φ is a homomorphism from $CD(G', P')$ onto $CD(G, P)$ since it is a cover. Thus φ must preserve regularity. But the domain of φ is regular by Claim 1 and its range is not regular by Claim 2. Thus φ cannot exist. So G' cannot cover G under any choice of φ and $H' \subseteq P'$. ▮

We now embark on the proof of another negative result by exhibiting a grammar which cannot be covered by any grammar in Greibach form. Thus the elimination of left recursive changes the structure of a grammar sufficiently that it cannot have a covering grammar.

THEOREM 1.4. *Let G be the following context-free grammar:*

$$S \to S0|S1|0|1$$

There is no grammar $G' = (V', \Sigma', P', S')$ in Greibach normal form such that (G', H') covers (G, P) under φ for any $H' \subseteq P'$ and φ mapping H' into P.

PROOF. The proof is by contradiction. Suppose there is a grammar

$$G' = (V', \Sigma', P', S')$$

in Greibach form such that G' is reduced and there exist $H' \subseteq P'$ and φ so that (G', H') covers (G, P) under φ.

CLAIM 1. $H' = P' \subseteq (N') \times (\Sigma N'^*)$.

PROOF. Suppose $x \in \Sigma^+$ and $S' \underset{R}{\overset{*}{\Rightarrow}} x$ in G' with canonical derivation $\pi = (\pi_1, \cdots, \pi_n)$. The H'-sparse derivation of π, π' has the property that $\varphi(\pi')$ is a P-sparse derivation of x in G. But then $\varphi(\pi')$ is a derivation of x in G. Since each rule of P contributes exactly one terminal character to x, and since $\varphi(\pi')$ is a derivation of $\lg(x)$ steps, $n \geq \lg(x)$. Since each π_i in P' contributes at least one terminal character to x, $\lg(x) \geq n$. Thus $n = \lg(x)$ and $\pi_i = \pi_i'$. Since G' is reduced it follows that $H' = P'$ and each $\pi_i \in P'$ contains exactly one terminal character. So since G' is in Greibach normal form $P' \subseteq N' \times (\Sigma N'^*)$.

Since every production in P' contains exactly one terminal character, the following result holds.

[8] Let $G = (V, \Sigma, P, S)$ be any grammar and let $a \in V$ and $x \in V^*$. We write $\#_a(x)$ for the number of occurrences in a in x. For any $i \geq 0$ and any $x = a_1 \cdots a_n$, $a_i \in \Sigma$ for $1 \leq i \leq n$, if $i \geq n$ then ${}^{(i)}x = x^{(i)} = x$. If $i < n$ then ${}^{(i)}x = a_1 \cdots a_i$ and $x^{(i)} = a_{n-i+1} \cdots a_n$.

CLAIM 2. *For any A in N', x in $(V')^*$, $A \overset{n}{\underset{R}{\Rightarrow}} x$ in G' implies*[9] $\#_\Sigma(x) = n$.

CLAIM 3. *For each $x \in (V')^*$, $A \in N'$, and $z \in \Sigma^*$; if $S' \overset{*}{\underset{R}{\Rightarrow}} xAz$ in G' and $\#_\Sigma(x) = k$ then there is a $y_A \in \Sigma^k$ so that*

$$\Sigma^*\{u \in \Sigma^* | A \overset{*}{\underset{G'}{\Rightarrow}} u\} \subseteq \Sigma^*\{y_A\}.$$

PROOF. Let $S' \overset{*}{\underset{R}{\Rightarrow}} xAz$ in G' by canonical derivation $(\pi_i)_{i=1}^n$. Now $\varphi(\pi_i)_{i=1}^n$ is a generation in G of Sw for some $w \in \Sigma^*$. First note that $\lg(w) = n$ since each production $\varphi(\pi_i)$ contributes exactly one character to w. Also note by Claim 2 that $n = \#_\Sigma(xAz) = \#_\Sigma(x) + \#_\Sigma(A) + \#_\Sigma(z) = k + \lg(z)$. So $\lg(w) = \lg(z) + k$. Now suppose $xA \overset{*}{\underset{R}{\Rightarrow}} u \in \Sigma^+$ in G' by $(\pi_i)_{i=n+1}^m$. Then $S' \overset{*}{\underset{R}{\Rightarrow}} uz$ in G' by $(\pi_i)_{i=1}^m$ and $\varphi(\pi_i)_{i=1}^m$ is a derivation $S \overset{*}{\underset{R}{\Rightarrow}} u'w = uz$ in G. So since[10] $\lg(w) = \lg(z) + k$, $w = (u^{(k)})z$. Thus w uniquely determines $u^{(k)}$. So Claim 3 is established.

Armed with this result we are in a position to complete the proof of the Theorem. Since $L(G')$ is not finite, there exists an $A \in N'$ such that $A \overset{+}{\underset{R}{\Rightarrow}} xAy$ for some $x \in V^*, y \in \Sigma^*$. Let $n = \lg(x)$. Since G' is in Greibach normal form, $n > 0$. Let $z \in \Sigma^*$ be the shortest terminal string generated by A in G', i.e. $A \overset{*}{\Rightarrow} z$ in G' and if $z' \in \Sigma^*$ and $A \overset{*}{\Rightarrow} z'$ in G' then $\lg(z) \leq \lg(z')$. z exists because G' is reduced. Let $m = \lg(z)$ and observe that $A \overset{*}{\Rightarrow} x^{m+1}Ay^{m+1}$ in G'. Since G' is reduced, there exist t, t' so that $S' \overset{*}{\underset{R}{\Rightarrow}} tAt' \overset{*}{\underset{R}{\Rightarrow}} tx^{m+1}Ay^{m+1}t' \overset{*}{\underset{R}{\Rightarrow}} tx^{m+1}zy^{m+1}t'$ in G'.

By Claim 3 and the fact that

$$\#_\Sigma(tx^{m+1}) \geq \#_\Sigma(x^{m+1}) = n(m+1),$$

we conclude that there exists a $y_A \in \Sigma^{n(m+1)}\Sigma^*$ such that

$$\Sigma^*\{z\} \subseteq \Sigma^*\{y_A\},$$

which implies that

$$z \in \Sigma^*\Sigma^{n(m+1)}\Sigma^* \subseteq \Sigma^*\Sigma^{n(m+1)}.$$

Thus

$$\lg(z) \geq n(m+1) \geq m+1 > m = \lg(z).$$

The contradiction indicates that the assumption that G' exists was fallacious. ∎

We now turn to the study of an important property of grammars used in programming language description.

Definition. A context-free grammar $G = (V, \Sigma, P, S)$ is said to be *invertible* if $A \rightarrow w$ and $B \rightarrow w$ in P implies $A = B$.

This property is very important in some bottom-up parsing schemes because once a simple phrase of a sentential form in an invertible grammar has been found, then the left-hand side of the production is uniquely and simply found.

Our first result says that for any grammar, there is an equivalent invertible grammar. This theorem was independently discovered by Graham [17, 18].

THEOREM 1.5. *For each context-free grammar $G = (V, \Sigma, P, S)$ there is an invertible context-free grammar $G' = (V', \Sigma, P', S')$ so that $L(G') = L(G)$. Moreover, if G is Λ-free then so is G'.*

PROOF. Let us assume, without loss of generality, that $G = (V, \Sigma, P, S)$ is

[9] Let $\#_\Sigma(x) = \sum_{a \in \Sigma} \#_a(x)$ so $\#_\Sigma(x)$ is the number of occurrences of terminals in x.

[10] Recall that $u^{(k)}$ is the suffix of u of length k.

Λ-free and chain-free. (If $\Lambda \in L(G)$ then $L_1 = L(G) - \{\Lambda\}$ has a grammar $G' = (V', \Sigma, P', S')$ which is Λ-free and chain-free. If the result has been proven for G', then take $G'' = (V' \cup \{S''\}, \Sigma, P'', S'')$ where $P'' = P' \cup \{S'' \to \Lambda, S'' \to S'\}$. Clearly $L(G'') = L(G)$ and G'' will be invertible if G' is.)

Let $G' = (V', \Sigma, P', S')$ where $N' = \{U \subseteq N \mid U \neq \varnothing\} \cup \{S'\}$ and S' is a new symbol not in V.

Thus the variables of G' (except S') will be nonempty subsets of the variables of G.

P' is defined as follows:

(a) $S' \to A$ where $S \in A \subseteq N'$ is in P'.

(b) For each production $B \to x_0B_1x_1 \cdots B_nx_n$ in P with $B_1, \cdots, B_n \in N$ and $x_0, \cdots, x_n \in \Sigma^*$, then for each $A_1, \cdots, A_n \in N' - \{S'\}$, P' contains

$$A \to x_0A_1x_1 \cdots A_nx_n$$

where

$$A = \{C \mid C \to x_0C_1x_1 \cdots C_nx_n \text{ is in } P \text{ for some } C_1, \cdots, C_n \text{ with each } C_i \in A_i\}.$$

If $C \to y_0C_1y_1 \cdots C_ny_n$ with $y_0, \cdots, y_n \in \Sigma^*$, $C_i \in N$, we call the string $y_0\text{-}y_1\text{-}\cdots\text{-}y_n$ the *stencil* of the production (variables replaced by dashes).

Note that P and P' have the same set of stencils and that G' is invertible. Assume without loss of generality that G' is reduced.

Before embarking on a proof that $L(G') = L(G)$, we give an example of the construction.

Example. Consider the following grammar:

$$S \to 0A|1B$$
$$A \to 0A|0S|1B$$
$$B \to 1|0$$

Applying the construction of the theorem leads to the following grammar.

$$\{B\} \to 1|0$$
$$\{A\} \to 0\{S\}|0\{S,B\}$$
$$\{A,S\} \to 0\{A\}|0\{A,B\}|0\{A,S\}|0\{A,S,B\}|1\{B\}|1\{B,A\}|1\{B,A,S\}|1\{B,S\}$$
$$S' \to \{S\}|\{A,S\}|\{B,S\}|\{A,B,S\}$$

Reducing the grammar leads to:

$$S' \to \{A,S\}$$
$$\{B\} \to 1|0$$
$$\{A,S\} \to 0\{A,S\}|1\{B\}$$

This is the familiar "subset construction" from automata theory [11].

Now we begin the proof that $L(G') = L(G)$.

CLAIM 1. *For each $A \in N'$ and each $x \in \Sigma^*$, $A \overset{*}{\Rightarrow} x$ in G' implies $B \overset{*}{\Rightarrow} x$ in G for each $B \in A$.*

PROOF. The argument is an induction on l, the length of a derivation in G'.

Basis. Suppose $l = 1$. Then $A \Rightarrow x \in \Sigma^*$ in G' and $A \rightarrow x$ is in P'. By the construction $A = \{C \in N \mid C \rightarrow x \text{ is in } P\}$. Clearly this holds if and only if $B \rightarrow x$ is in P for each $B \in A$.

Induction Step. Suppose $l \geq 2$ and Claim 1 holds for all derivations of length less than l. Then suppose $A \Rightarrow x_0A_1x_1 \cdots A_nx_n \overset{*}{\Rightarrow} x$ in G' by a derivation of length l. This implies that for each i, $1 \leq i \leq n$, $A_i \overset{*}{\Rightarrow} y_i \in \Sigma^*$ in G' and $x_0y_1x_1 \cdots y_nx_n = x$. By the construction, for each $B \in A$ there exist $B_i \in A_i$ so that $B \rightarrow x_0B_1x_1 \cdots B_nx_n$ is in P. Moreover, the induction hypothesis implies that $B_i \overset{*}{\Rightarrow} y_i$ in G and therefore

$$B \Rightarrow x_0B_1x_1 \cdots B_nx_n \overset{*}{\Rightarrow} x_0y_1x_1 \cdots y_nx_n = x \text{ in } G.$$

Note that Claim 1 implies that $L(G') \subseteq L(G)$.

To complete the proof, the following result is needed.

CLAIM 2. *For each $x \in \Sigma^*$, let $X = \{C \in N \mid C \overset{*}{\Rightarrow} x \text{ in } G\}$. If $B \overset{*}{\Rightarrow} x$ in G then $A \overset{*}{\Rightarrow} x$ in G' for some A such that $B \in A \subseteq X$.*

PROOF. The argument is an induction on l, the length of a derivation in G.

BASIS. $l = 1$. Suppose $B \Rightarrow x$ in G so $B \rightarrow x$ is in P. Then by construction $A \rightarrow x$ is in P' with $B \in A = \{C \in N \mid C \rightarrow x \text{ is in } P\}$.

Induction Step. Suppose $B \Rightarrow x_0B_1x_1 \cdots B_nx_n \overset{*}{\Rightarrow} x_0y_1x_1 \cdots y_nx_n = x \in \Sigma^*$ in G is a derivation of length l. There are derivations $B_i \overset{*}{\Rightarrow} y_i$, all of which have length less than l. By the induction hypothesis, there are $A_i \in N'$ so that for each i, $A_i \overset{*}{\Rightarrow} y_i$ in G', and $B_i \in A_i$. By the construction $A \rightarrow x_0A_1x_1 \cdots A_nx_n$ is in P' with $B \in A$. Thus $A \Rightarrow x_0A_1x_1 \cdots A_nx_n \overset{*}{\Rightarrow} x_0y_1x_1 \cdots y_nx_n = x$ in G'.

By Claim 2, $L(G') \supseteq L(G)$ and hence $L(G') = L(G)$. ∎

It is easy to see that the invertibility condition is compatible with conditions (a) through (e) and not compatible with (f) in the Introduction. It is interesting to note that for any grammar G, one can find an equivalent grammar G' which is invertible and chain-free. On the other hand, there are grammars G for which there do not exist equivalent grammars which are invertible, chain-free, and Λ-free. An example of such a grammar is:

$$S \rightarrow A|b$$

$$A \rightarrow aA|a$$

(To prove this, suppose that G' is such a grammar. One can easily show by induction that for each $i \geq 1$, $a^i \in L(G')$ implies $S \rightarrow a^i$ is in P'. For $L(G')$ to equal $L(G)$ it must follow that P' is infinite which is a contradiction.)

The grammar G' of Theorem 1.5 does not necessarily cover G. For example, if G is the grammar:

$$S \rightarrow A|B$$

$$A \rightarrow a$$

$$B \rightarrow a$$

then G' is:

$$\{S\} \rightarrow \{A,B\}$$

$$\{A,B\} \rightarrow a$$

which cannot cover G since φ must be a function. However the grammar:

$$S \to A|B$$
$$A \to a$$
$$B \to aL$$
$$L \to \Lambda$$

does completely cover G. Generalizing this result we obtain the following theorem.

THEOREM 1.6. *Let $G = (V, \Sigma, P, S)$ be a Λ-free context-free grammar. Then G is completely covered by an invertible grammar G'.*

PROOF. We simply present the construction. Index the elements of N by the integers[11] $1, 2, \cdots, |N|$. Let the index of $A \in N$ be denoted $I(A)$. Let L be a new symbol and construct $G' = (V', \Sigma, P', S)$ as follows:

$$N' = N \cup \{L\}$$
$$P' = \{A \to xL^i \mid A \to x \in P \text{ and } I(A) = i\} \cup \{L \to \Lambda\}$$

Then (G', H) covers (G, P) under φ where $H = P' - \{L \to \Lambda\}$ and where $\varphi: H \to P$ is defined by $\varphi(A \to xL^i) = (A \to x)$ for each $A \in N$, $i = I(A)$, $(A \to xL^i) \in H$. ∎

It is easy to see that the grammar:

$$S \to A|B$$
$$A \to a$$
$$B \to a$$

cannot be completely covered by any invertible grammar which is Λ-free.

These results indicate theoretical applications of covers. It should be noted that Theorem 1.5 is a generalization of a result by McNaughton [14] on parenthesis grammars. The difference between Theorems 1.5 and 1.6 is quite illuminating. Theorem 1.6 does give a covering while Theorem 1.5 does not. On the other hand, the construction of Theorem 1.6 leads to a resulting grammar G' which has Λ-rules even when G does not.

Although the construction given in Theorem 1.2 uses Λ-rules in a similar way, null rules can be eliminated by a more complex construction; cf. the remarks following Theorem 1.2.

Theorems 1.2 and 1.4 are quite surprising in a number of ways. First it is surprising to be able to prove that the Greibach normal form (elimination of left recursion) alters parse trees so significantly that no covering grammar can exist. (This is as much of a consequence of our definition of covering as it is of the normal form.) In light of Theorem 1.4, Theorem 1.2 is even more surprising. The previous operator normal form construction [10] had first constructed the Greibach normal form of the grammar and then gone to an operator form. Theorem 1.4 shows that such transformations can never be expected to lead to a covering, but we have seen that a simple direct construction will work for Λ-free grammars.

2. *Bottom-Up Parsing*

Bottom-up parsing methods are usually described as algorithms which scan an input stream while computing with a pushdown store and a bounded amount of additional

[11] For any set X, the cardinality of X is denoted by $|X|$.

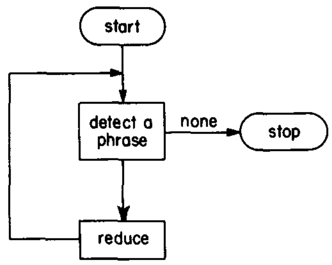


FIG. 3. Flowchart of a bottom-up parser.

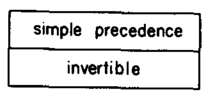
memory. At each stage, the algorithm performs one of the following actions:

(1) reads an input symbol onto the stack; this continues until a complete phrase resides in the stack; or

(2) replaces the phrase in the stack by a nonterminal which generated it.

The first action is called *phrase detection* while the second operation is called *phrase reduction*. The entire algorithm can be represented by the flowchart shown in Figure 3.

For example, Wirth and Weber [16] present a bottom-up parsing scheme for invertible simple precedence grammars.[12] They do reduction using dictionary lookup and they detect phrases using simple precedence relations. Based on the model of Figure 3, this type of parser can be represented by the following diagram:



where the upper box indicates the detection method while the lower box represents the reduction scheme. It is known [3] that the above class (simple precedence detection and invertible reduction) is not powerful enough to parse all context-free languages. In our more general framework, it is natural to inquire about the potency of simple precedence detection and of invertible reduction for particular grammars.

Is invertible reduction powerful enough to parse every context-free grammar? The answer to this question depends on one's notion of adequate. If one requires that every grammar be equivalent to an invertible grammar then the answer is yes by virtue of Theorem 1.5. In the previous section, we argued that adequacy is essentially the ability to cover, i.e. parsing G' is as good as parsing G if and only if G' covers G. If our definition of adequacy is that every grammar be completely covered by an invertible grammar then we must examine Theorem 1.6. We know that we can completely cover a Λ-free grammar G by an invertible grammar G'. But the proof of Theorem 1.6 reveals that although G is Λ-free, G' has null rules (and is more complicated to parse than G in that respect at least). We have already seen (cf. remarks after Theorem 1.6) that there are (Λ-free) grammars which cannot be completely covered by an invertible Λ-free grammar. In light of this, the answer to our original question can be taken to be no.

In some sense this means that the reduction phase of a general parser must be

[12] In this introduction, we will discuss a number of special types of grammars such as simple precedence grammars. Formal definitions occur in this paper before the mathematical use of each concept. Definitions for concepts which are discussed but are not used in theorems may be found in [8, 9].

nontrivial. Surprising enough we shall now show that all the "work" in bottom-up parsing can be done by the reduction phase.

What does it mean to shift all the work in parsing to the reduction phase? Since simple precedence is the simplest form of phrase detection, we ask whether every grammar may be completely covered by a simple precedence grammar. The (surprising (?)) answer is yes. In fact, we can say much more; we can cover grammars of type X by simple precedence grammars and reduce them by techniques appropriate for type X grammars. In particular, some of the results that we can prove are as follows:

simple precedence / LR(k)	completely covers [13]	LR(k) / LR(k)
simple precedence / bounded right context [14]	completely covers	bounded right context / bounded right context
simple precedence / unambiguous	completely covers	unambiguous / unambiguous
simple precedence / nondeterministic	completely covers	nondeterministic / nondeterministic

Each of the above results is in [8].

Lest the reader try to formulate the theorem "for all X, every grammar of type X can be covered by a grammar which is precedence detectable and X reducible" we point out that not every invertible grammar is covered by a precedence detectable invertible grammar. To see this observe that precedence detection plus invertibility cannot handle all bounded right context languages [3, 13]. On the other hand Theorem 1.6 shows that every context-free language has an invertible grammar.

In order to prove our main results, we need some additional concepts.

Definition. A context-free grammar $G = (V, \Sigma, P, S)$ is said to be *chain reduced* if G is reduced and if for any $A \in N$ it is not the case that $A \overset{+}{\Rightarrow} A$.

If a grammar is not chain reduced then it is ambiguous. One can easily decide whether a grammar is chain reduced and if it is not, one can remove the "cycles" by a straightforward construction and then reduce it. Note that a chain reduced grammar may have chains but they are of bounded length.

Before we can state the next result, we must recall the formalism for LR(k) grammars [8] and assume that none of our grammars contain the rule $S \rightarrow S$.

Definition. Let k be any positive integer. The grammar $G = (V, \Sigma, P, S)$ is called LR(k) *detectable* if for any $x, y, y' \in V^*$; $A, A' \in N$; $z, z' \in \Sigma^*$ if $S \overset{*}{\underset{R}{\Rightarrow}} xyz$ has handle $(A \rightarrow y, \lg(xy))$ and $S \overset{*}{\underset{R}{\Rightarrow}} xyz'$ has handle $(A' \rightarrow y', j)$ and ${}^{(k)}z = {}^{(k)}z'$ then $j = \lg(xy)$ and $y' = y$.

[13] This notation is an informal way to state the theorem that every bounded right context grammar G is covered by a grammar G' which is simple precedence detectable and bounded right context reducible.

[14] See Footnote 13; see [8].

Note that $A' = A$ is not necessarily true.

Definition. G is said to be LR(k) *reducible* if (under the same quantification as above) whenever $S \underset{R}{\overset{*}{\Rightarrow}} xyz$ has handle $(A \to y, \lg(xy))$ and $S \underset{R}{\overset{*}{\Rightarrow}} xyz'$ has handle $(A' \to y, \lg(xy))$, then $A = A'$.

G is said to be LR(k) if it is LR(k) detectable and LR(k) reducible, i.e. if $S \underset{R}{\overset{*}{\Rightarrow}} xyz$ has handle $(A \to y, \lg(xy))$ and $S \underset{R}{\overset{*}{\Rightarrow}} xyz'$ has handle $(A' \to y', j)$ and ${}^{(k)}z = {}^{(k)}z'$ then $(A \to y, \lg(xy)) = (A' \to y', j)$.

Our next result, while interesting in its own right, is intended as a device to help prove Theorem 2.2.

THEOREM 2.1. *Every* LR(k) *grammar G can be completely covered by an* LR(k) *canonical two form grammar G'. If G is chain reduced and Λ-free so is G'.*

PROOF. We will invoke the construction of Theorem 1.1 to define G'. By Theorem 2.1 G' covers G. Inspection of P' shows that if G is chain reduced and Λ-free then so is G'. It remains to be seen that G' is LR(k) if G is.

Assume that G is LR(k).

LEMMA 1. *Let xyz be a canonical sentential form of G' with handle $(A \to y, lg(xy))$. Let the canonical sentential form*[15] *$\varphi(xyz)$ have handle $(B \to v, m)$ in G. Then*

(*a*) $xyz \in (N')^*\Sigma^*$,

(*b*) if $A \to y$ *is in* $P_1 \cup P_2 \cup P_3$ *then* $m = lg(\varphi(xy))$ *and* $lg(\varphi(y)) = lg(v)$,

(*c*) *if* $A \to y$ *is in* P_4 *then* $m = lg(\varphi(xy))$ *and* $lg(\varphi(y)) < lg(v)$,

(*d*) *if* $A \to y$ *is in* P_5 *then* $m \geq lg(\varphi(xy))$.

PROOF. We induct on the minimal n such that $[S] \underset{R}{\overset{n-1}{\Rightarrow}} xAz \underset{R}{\Rightarrow} xyz$ in G'.

Basis. $n = 1$ implies $A = [S]$ and inspection of P' shows that $A \to y$ is in $P_1 \cup P_2 \cup P_3$. Since φ is a cover, $\varphi(xyz)$ has handle $(S \to \varphi(y), \lg(\varphi(xy)))$ and so (a) and (b) are established while (c) and (d) hold vacuously.

Induction Step. We proceed by cases. If $A \to y$ is in $P_1 \cup P_2 \cup P_3$ the above logic is still valid. If $A \to y$ is in P_4, then by inspection of P_4, $A = [A_1 \cdots A_q]$ for some $q \geq 2$, $A_1, \cdots, A_q \in V$, and $y = [A_1][A_2 \cdots A_q]$. By hypothesis $\varphi(xAz)$ has handle $(B \to v, \lg(\varphi(xA)))$, and $\lg(v) > \lg(\varphi(A))$. So since $\varphi(A) = \varphi(y)$, $\lg(v) > \lg(\varphi(A)) = \lg(y)$ and (c) holds. But $m = \lg(\varphi(xA)) = \lg(\varphi(xy))$ because $\varphi(A) = \varphi(y)$. Thus (a) and (c) hold and (b) and (d) are vacuous. Lastly if $A \to y$ is in P_5, then by the induction hypothesis $m \geq \lg(\varphi(xA))$ and by inspection of P_5, $\lg(\varphi(A)) = \lg(\varphi(y))$. Thus $m \geq \lg(\varphi(xy))$. Thus (a) and (d) follow and (b) and (c) are vacuously satisfied and the lemma follows by induction.

Now we must prove that G' is LR(k). Suppose that for any $x, y, y' \in (V')^*$; $z, z' \in \Sigma^*$; xyz is a canonical sentential form of G' with handle $(A \to y, j)$ where $j = \lg(xy)$, and xyz' is a canonical sentential form of G' with handle $(A' \to y', j')$, and ${}^{(k)}z = {}^{(k)}z'$. Then we must show that $(A \to y, j) = (A' \to y', j')$.

The proof now breaks into cases. We first deal with the case in which $A \to y$ is in P_5. In that case, we will show that the handles are equal. Then, under the assumption that $A \to y$ is not in P_5, we must consider subcases as to which P_i the production $A \to y$ belongs. In each subcase, we show the handles are equal.

Case 1. $A \to y$ is in P_5. Inspection of P_5 shows $y \in \Sigma$. Suppose $A' \to y'$ is not in P_5. Then by Lemma 1 ${}^{(j')}(xyz') \subseteq N'^*$. Hence $j' < \lg(xy)$. Note that ${}^{(j'+k)}(xyz) = {}^{(j'+k)}(xyz')$ since ${}^{(k)}z = {}^{(k)}z'$. Let $\varphi(xyz)$ have handle $(B \to v, m)$ in G and let

[15] We also write $\varphi(x)$ for $x \in (V')^*$ although φ was initially defined on productions. It is actually a homomorphic extension of the function $\varphi(a) = a$ for $a \in \Sigma$ and $\varphi([A]) = A$ for $[A]$ in N'.

$\varphi(xyz')$ have handle $(B' \to v', m')$ in G By Lemma 1, $m \geq \lg(\varphi(xy))$ and $m' = \lg(\varphi(^{(j')}(xyz')))$. Since $^{(j'+k)}(xyz) = {}^{(j'+k)}(xyz')$ it follows that $^{(m'+k)}(\varphi(xyz)) = {}^{(m'+k)}(\varphi(xyz'))$. Invoking the hypothesis that G is LR(k) yields $(B \to v, m) = (B' \to v', m')$ is the handle of both $\varphi(xyz)$ and $\varphi(xyz')$. In particular $m' = m$. However, it was established above that $j' < \lg(xy)$ so $m' = \lg(\varphi(^{(j')}(xyz'))) \leq \lg(\varphi(x)) < \lg(\varphi(xy)) \leq m$. So $m' < m$. This contradiction shows $A' \to y'$ is in P_5. A symmetric argument shows that $A \to y$ is in P_5 if $A' \to y'$ is. So we conclude that $A \to y$ is in P_5 if and only if $A' \to y'$ is.

Next observe that in this case $y, y' \in \Sigma$. In particular y and y' are the leftmost terminal characters of xyz and xyz' respectively. So $y = y'$, $j = j'$ and by inspection of P_5, $A = A' = [y]$. This establishes that $(A \to y, j) = (A' \to y', j')$ if $A \to y$ is in P_5.

Case 2. $A \to y$ is not in P_5. In the above case we concluded $A \to y$ is in P_5 if and only if $A' \to y'$ is in P_5. So we observe that $A' \to y'$ is not in P_5 in this case. Now by (b) and (c) of Lemma 1, if $\varphi(xyz)$ has handle $(B \to v, m)$ in G then $m = \lg(\varphi(xy))$. So since $^{(k)}z = {}^{(k)}z' = {}^{(k)}\varphi(z) = {}^{(k)}\varphi(z')$ and since G is LR(k) we conclude that $\varphi(xyz')$ has handle $(B \to v, m)$ in G. By (b) and (c) of Lemma 1, this means that $m = \lg(\varphi(^{(j')}(xyz')))$. So $\varphi(^{(j')}(xyz')) = \varphi(xy)$. This means $j' = \lg(xy)$ so $j' = j$.

To summarize the assumptions and conclusions of the above paragraph:

(i) xyz has handle $(A \to y, \lg(xy))$ in G' and $A \to y$ is not in P_5,

(ii) xyz' has handle $(A' \to y', \lg(xy))$ in G' and $A' \to y'$ is not in P_5,

(iii) $^{(k)}(z) = {}^{(k)}(z')$,

(iv) $\varphi(xyz)$ and $\varphi(xyz')$ both have handle $(B \to v, \lg(\varphi(xy)))$ in G.

Now the argument divides into subcases.

Case 2.1. $A \to y$ is in $P_1 \cup P_2 \cup P_3$. If $A' \to y'$ is in $P_1 \cup P_2 \cup P_3$ then since φ is a cover $(B \to v) = \varphi(A \to y) = \varphi(A' \to y')$. Inspection of $P_1 \cup P_2 \cup P_3$ and φ shows that in this case $(A \to y) = (A' \to y')$. If $A' \to y'$ is in P_4 then by Lemma 1(c) $\lg(v) > \lg(\varphi(y'))$. But $\lg(\varphi(y')) \geq \lg(\varphi(y))$ and since $\varphi(A \to y) = (B \to v)$ it follows that $\lg(v) > \lg(\varphi(y')) \geq \lg(\varphi(y)) = \lg(v)$. This contradiction shows that $A' \to y'$ is not in P_4. Hence it is in $P_1 \cup P_2 \cup P_3$ and therefore $(A \to y) = (A' \to y')$.

Case 2.2. $A \to y$ is in P_4. By symmetry the above arguments require that $A \to y$ is not in P_4 if $A' \to y'$ is not in P_4. So $A' \to y'$ is in P_4. Inspection of P_4 shows that $(A \to y) = (A' \to y')$ in this case.

The above arguments have shown that in any case $(A \to y, j) = (A' \to y', j')$. So it follows that G' is LR(k). ▌

Before stating our main result, we need the following concepts about precedence analysis. The reader is referred to [9] which presents our theory in greater detail and generality.

Definition. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar with delimiter.[16] Define the following binary relations on V:

$$\lambda = \{(A,B) \mid A \to By \text{ is in } P \text{ for some } y \in V^*\},$$
$$\rho = \{(A,B) \mid B \to xA \text{ is in } P \text{ for some } x \in V^*\},$$
$$\alpha = \{(A,B) \mid C \to xABz \text{ is in } P \text{ for some } x, z \in V^*\} \cup \{(\perp,S), (S,\perp)\}.$$

[16] At this point, we use context-free grammars with delimiters. Formally, the conventions are that $\perp \in \Sigma$, $\perp S \perp$ is the start string, and $P \subseteq (V - \Sigma) \times (V - \{\perp\})^*$. All of the previous theorems are true with minor modifications for grammars with delimiters (cf. [8]).

Finally, define

$\lessdot = \alpha\lambda^+,$
$\doteq = \alpha,$
$\gtrdot = (\rho^+\alpha\lambda^*) \cap (V \times \Sigma).$

The reader who is familiar with the general theory of canonical precedence will note that this is the special case where $T = V$ so that G is Λ-free, $\alpha = \gamma$, $\lambda = \delta$, and $\rho = \omega$.

Now we can give the following definition.

Definition. A context-free grammar $G = (V, \Sigma, P, \perp S \perp)$ is said to be a *precedence detectable grammar* if

(a) G is Λ-free, and
(b) the relations $\lessdot$, $\doteq$, and $\gtrdot$ are pairwise disjoint.

We can now state and prove the main result of this section.

THEOREM 2.2. *If G is a Λ-free chain reduced* LR(k) *grammar in canonical two form, then G is completely covered by a simple precedence detectable,* LR(k) *reducible grammar G'.*

PROOF. Let $G = (V, \Sigma, P, \perp S \perp)$ be a chain reduced Λ-free LR(k) canonical two form grammar. For each $A \in N$ define $p(A) = \max \{m \mid A_0, \cdots, A_m \in N; A = A_0 \Rightarrow A_1 \Rightarrow \cdots \Rightarrow A_m\}$. Since G is chain reduced, $p(A)$ exists and is bounded by $0 \leq p(A) \leq |N|$. Further if $A \Rightarrow B$ then $p(A) > p(B)$. Define $p = \max_{A \in N} p(A)$ for G. Now define $G' = (V', \Sigma, P', \perp S \perp)$ where

$$V' = \{[A,i] \mid 0 \leq i \leq p + 2, A \in N\} \cup \{S\} \cup \Sigma.$$

Let

$P_1 = \{S \rightarrow [S,p+2]\},$
$P_2 = \{[A,p+2] \rightarrow [A,p] \mid A \in N\},$
$P_3 = \{[A,p+1] \rightarrow [A,p] \mid A \in N\},$
$P_4 = \{[A,i] \rightarrow [A,i-1] \mid A \in N; 0 < i \leq p\},$
$P_5 = \{[A,0] \rightarrow a \mid A \in V-\Sigma, a \in \Sigma; A \rightarrow a \text{ in } P\},$
$P_6 = \{[A,p(A)] \rightarrow [B,p(B)] \mid A, B \in N; A \rightarrow B \text{ in } P\}$, and
$P_7 = \{[A,0] \rightarrow [B,p+2][C,p+1] \mid A, B, C \in N; A \rightarrow BC \text{ in } P\}.$

Let $P' = \bigcup_{i=1}^{7} P_i$. Let $H' = \bigcup_{i=5}^{7} P_i$. Now define $\varphi : V' \rightarrow V$ by

$$\varphi(a) = a \quad \text{for each } a \in \Sigma,$$
$$\varphi(S) = S,$$
$$\varphi([A,i]) = A \quad \text{for each } A \in N,\ 0 \leq i \leq p + 2,$$

extend φ to a homomorphism of $(V')^*$ onto V^*, and define φ on H by

$$\varphi(A \rightarrow x) = \varphi(A) \rightarrow \varphi(x) \text{ for each } A \rightarrow x \text{ in } H.$$

This construction is rather complex. The reader should note that if $p = 0$ this is essentially the construction of Fischer [3]. The need for p stems from the necessity of covering chain productions and hence a need to bracket each nonterminal by $\lessdot$ and $\gtrdot$ at most p times. To establish the theorem one must show

(a) (G', H') covers G under φ,
(b) G' is LR(k),
(c) G' is simple precedence detectable.

The techniques for the proof of (a) are presented in Theorem 1.1; the technique for the proof of (b) is presented in Theorem 2.1. For the sake of brevity we omit these proofs and prove only (c).

CLAIM. *G' is a simple precedence grammar.*

PROOF. It suffices to show that $\lessdot_V$, $\doteq_V$, and $\gtrdot_V$ are pairwise disjoint. Inspection of P' shows the following:

$$\alpha \subseteq \{(\perp,S),(S,\perp)\} \cup (\{[A,p+2] \mid A \in N\} \times \{[A,p+1] \mid A \in N\})$$

$$\lambda^+ \subseteq (\{S\} \times (V' - \{\perp,S\})) \cup (\{[A,i] \mid A \in N; 0 \leq i \leq p+2\} \times (\{[A,i] \mid A \in N; 0 \leq i \leq p+2, i \neq p+1\} \cup (\Sigma - \{\perp\}))$$

$$\rho^+ \subseteq ((V' - \{\perp,S\}) \times \{S\}) \cup ((\{[A,i] \mid A \in V-\Sigma, 0 \leq i \leq p+2, i \neq p+2\} \cup (\Sigma - \{\perp\})) \times \{[A,i] \mid A \in N, 0 \leq i \leq p+2\})$$

so

$$\lessdot\ = \alpha\lambda^+ \subseteq (\{\perp\} \times (V' - \{\perp,S\}) \cup (\{[A,p+2] \mid A \in N\} \times (\{[A,i] \mid A \in N, 0 \leq i \leq p+2, i \neq p+1\} \cup (\Sigma - \{\perp\}))$$

$$\doteq\ = \alpha$$

$$\gtrdot\ = \rho^+\alpha\lambda^* \cap (V' \times \Sigma) \subseteq ((V' - \{\perp,S\}) \times \{\perp\}) \cup ((\{[A,i] \mid A \in N, 0 \leq i \leq p+2, i \neq p+2\} \cup (\Sigma - \{\perp\})) \times (\Sigma - \{\perp\}))$$

So the relations are disjoint. We display this result by the table:

	$\perp$	$\Sigma - \{\perp\}$	$[A,p+2]$	$[A,p+1]$	$[A,i]$	S
$\perp$		$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$
$\Sigma - \{\perp\}$	$\gtrdot$	$\gtrdot$				
$[A,p+2]$		$\lessdot$	$\lessdot$	$\doteq$	$\lessdot$	
$[A,p+1]$	$\gtrdot$	$\gtrdot$				
$[A,i]$	$\gtrdot$	$\gtrdot$				
S	$\doteq$					

Combining these results leads immediately to our main theorem.

THEOREM 2.3. *Every Λ-free chain reduced* LR(k) *grammar is completely covered by a precedence detectable,* LR(k) *reducible grammar.*

PROOF. The result follows immediately from Theorem 2.1, Theorem 2.2, and the transitivity of covers. ∎

By analogous techniques, one can show the following result.

THEOREM 2.4. *Every Λ-free chain reduced BRC(n, m) grammar is completely covered by a precedence detectable, BRC ($n + r$, m) reducible grammar for some integer r.*

Graham [17, 18] has independently proved that for any Λ-free LR(k) grammar (respectively BRC(n, m)) there is an equivalent LR(k) grammar (respectively BRC(n', m)) grammar with pairwise disjoint simple precedence relations.

3. *Summary and Conclusions*

Past work in the areas of normal forms and of classes of parsers has focussed primarily on the existence of a certain normal form for a grammar or the existence of a recognizer for a language. Often the proof is by a construction which mutilates the structure of the original grammar or produces an impractically large grammar. In an attempt to define and examine these properties one is led to the concept of grammatical covering. The definition of covering, although intuitively quite simple, is formally complex and gives rise to rather lengthy proofs. However, the definition yields some interesting results.

It shows, as expected, that the canonical two form is universal and that any conceivable Greibach normal form construction significantly changes the shape of the parse trees of some grammars. Surprisingly there exist constructions for the operator normal form which do not significantly change the shape and labeling of the parse trees. Similarly there exist constructions for the invertible form of a grammar which do not significantly change the shape of the parse trees.

Perhaps a word of caution is appropriate here. The constructions presented work as claimed. However, the resulting grammars are typically considerably larger than the original (Theorem 1.1 yields a EULER [16] grammar two times larger, Theorem 1.2 yields a grammar 1600 times larger, Theorem 1.5 yields a grammar 2^{40} times arger, and Theorem 2.2 yields a grammar 16 times larger). The theorems present ertain tricks which apply uniformly to the entire grammar. However, in practical ituations, they should be used incrementally and with discretion to repair local nomalies in a grammar. The substance of any particular theorem is that there is is not) hope of going from grammar G to a covering normal form grammar. Beyond hat, one is left pretty much to his own devices.

For example, Ichbiah and Morse [12] present a compact and fast parser which ses a precedence detection scheme and LR(k) reduction. Theorem 2.2 indicates hat their technique can handle all LR(k) grammars. By employing the construc-ons of Theorem 1.1 and Theorem 2.1 it is possible to convert any Λ-free and chain-ee LR(k) grammar G to a grammar G' which is precedence detectable and LR(k) educible. Further, this new grammar completely covers the original grammar. hus employing Figure 1.1 one can build a parser for G which uses precedence de-ection and LR(k) reduction on G' and translates G' parses to G parses by dictionary okup at each step of the parse.

EFERENCES

. CHOMSKY, N. Formal properties of grammars. In *Handbook of Mathematical Psychology, Vol. 2*, R. R. Bush, E. H. Galanter, and R. D. Luce (Eds.), Wiley, New York, 1962, pp. 323–418.

. EARLEY, J. "An efficient context free parsing algorithm," *Comm. ACM 13*, 2 (Feb. 1970), 94–102.

. FISCHER, M. J. Some properties of precedence languages. *Proc. Symp. on Theory of Computing*, May 1969, pp. 181–190.

. FLOYD, R. W. Syntactic analysis and operator precedence. *J. ACM 10*, 3 (July 1963), 316–333.

FLOYD, R. W. Bounded context syntactic analysis. *Comm. ACM 7*, 2 (Feb. 1964), 62–67.

GINSBURG, S. *The Mathematical Theory of Context Free Languages*. McGraw-Hill, New York, 1966.

GINSBURG, S., AND HARRISON, M. A. Bracketed context free languages. *J. Computer and System Sci. 1* (1967), 1–23.

GRAY, J. N. Precedence parsers for programming languages. Ph.D. Th., Dep. of Computer Sci., U. of California, Berkeley, Calif., Sept. 1969.

GRAY, J. N., AND HARRISON, M. A. Canonical precedence schemes. *J. ACM* (to appear).

GREIBACH, S. A. A new normal form theorem for context free phrase structure grammars. *J. ACM 12*, 1 (Jan. 1965), 42–52.

11. HOPCROFT, J. E., AND ULLMAN, J. D. *Formal Languages and Their Relation to Automata.* Addison Wesley, Reading, Mass., 1969.
12. ICHBIAH, J. D., AND MORSE, S. P. A technique for generating almost optimal Floyd-Evans productions for precedence grammars. *Comm. ACM, 13,* 8 (Aug. 1970), 501–508.
13. KNUTH, D. E. On the translation of languages from left to right. *Information and Control 8* (1965), 607–639.
14. MCNAUGHTON, R. Parenthesis grammars. *J. ACM, 14,* 3 (July 1967), 490–500.
15. REYNOLDS, J. C., AND HASKELL, R. Grammatical coverings. (unpublished manuscript, 1970).
16. WIRTH, N., AND WEBER, H. EULER: A Generalization of ALGOL and its Formal definition, Parts I, II. *Comm. ACM 9,* 1, 2 (Jan., Feb. 1966), 11–23, 89–99.
17. GRAHAM, S. L. Extended precedence, bounded right context languages, and deterministic languages (extended abstract). Proc. Symp. on Switching and Automata Theory, Oct. 1970, pp. 175–180.
18. GRAHAM, S. L. Precedence languages and bounded right context languages. Ph.D. Thesis, Dept. of Computer Sci., Stanford U., Stanford, Calif., July 1971.

RECEIVED JULY 1971; REVISED DECEMBER 1971