Pure Grammars

H. A. Maurer,* A. Salomaa,† and D. Wood‡

* Institut für Informationsverarbeitung, Technische Universität Graz, A-8010 Graz, Austria;
† Department of Mathematics, University of Turku, Finland; and
‡ Unit for Computer Science, McMaster University, Hamilton, Ontario, Canada L8S 4K1

This paper investigates generative grammars with only one type of symbol: no distinction is made between terminals and nonterminals. This means that all intermediate words in a derivation are necessarily in the language generated and, consequently, such languages differ considerably from languages generated by grammars, where nonterminals can be used to exclude words from the language. We investigate in this paper basic question concerning language hierarchies, ambiguity and decidability. Also some more general models of pure grammars will be considered.

1. INTRODUCTION

It has become customary in formal language theory to divide the alphabet of a grammar into two parts: nonterminal letters and terminal letters. Only words consisting entirely of terminal letters are considered to be in the language generated. Basically, this distinction stems from linguistic motivation: nonterminals represent syntactic classes of the languages.

On the other hand, such a distinction is not made in the original rewriting systems of Thue. Indeed it seems to us most natural to continue this original line of research and extend the considerations to concern, for instance, context-free rewriting. It is also possible that the investigation of such “pure” grammars will reveal properties of formal languages more important than those encountered in the study of grammars with terminals. The present paper is to be understood only as a starting point for this investigation: many of the most important problems are still open.

Pure grammars have been studied before in Gabrielian (1970) and Buttelmann et al. (1974). The main results of these two papers will be referenced below at appropriate places. The original work in L systems (cf. Salomaa (1968) and the references given there) was also carried out along the same lines: L systems were originally rewriting systems without nonterminals. Some work done in sentential forms of context-free grammars, cf. for instance Harju and Penttonen (1978) and Salomaa (1973b), is related to some of the problems discussed.
below. Finally, Ehrenfeucht and Rozenberg (1978) is a recent paper of interest to pure grammars. However, it is not directly related to the technical problems discussed below.

A brief outline of the contents of the present paper follows. After giving the basic definitions, we discuss in Section 3 the relation of some pure language families to the families in the Chomsky hierarchy. The next section is devoted to the case of one-letter alphabets, where practically everything can be settled. Section 5 deals with length sets: it turns out that some important hierarchy results can be established based on length considerations alone. The following section deals with problems involving regular languages: decision problems as well as characterization results for pure context-free grammars having a one-sided linear form. Some further decision problems are considered in Section 7. Section 8 discusses questions of ambiguity and monogenicity. In particular, we show the existence of a languages which cannot be generated monogenically by a context-free pure grammar but can be generated monogenically by a length-increasing pure grammar.

Finally, Section 9 outlines some possibilities for a more general set-up of pure grammars. The results in this section are mainly based on some rather old work in formal language theory.

2. BASIC DEFINITIONS

We expect the reader to be familiar with the basics of formal language theory. For all unexplained notions, we refer to Maurer (1969) or Salomaa (1973a).

The length of a word $w$ is denoted by $|w|$. By definition, the length of the empty word $\lambda$ equals 0.

For a language $L$, its length set is defined by

$$LS(L) = \{ |w| \mid w \in L \}.$$  

To avoid some trivial exceptional cases, we make the convention that two languages are considered equal if they differ by at most the empty word. Two language families are considered equal if they contain the same languages (modulo $\lambda$).

We now introduce the most important notions of this paper.

**Definition.** A pure grammar is a triple $G = (\Sigma, P, S)$ where $\Sigma$ is a finite alphabet, $S$ is a finite set of words over $\Sigma$, and $P$ is a finite set of ordered pairs $(x, y)$ of words over $\Sigma$. The elements of $P$ are referred to as productions and usually denoted by $x \rightarrow y$. 

We say that a word \( w \) over \( \Sigma \) yields directly a word \( w' \) over \( \Sigma \) according to \( G \), in symbols
\[
w \xrightarrow{G} w'
\]
or briefly \( w \Rightarrow w' \) (if \( G \) is understood) if there are words \( w_1 \) and \( w_2 \) and a production \( x \rightarrow y \) in \( P \) such that
\[
w = w_1 x w_2 \quad \text{and} \quad w' = w_1 y w_2.
\]
The reflexive transitive (resp. transitive) closure of the relation \( \Rightarrow \) is denoted by \( \overset{*}{\Rightarrow} \) (resp. \( \overset{+}{\Rightarrow} \)).

The language generated by \( G \) is defined by
\[
L(G) = \{ w \mid s \overset{*}{\Rightarrow} w, \text{ for some } s \in S \}.
\]
Languages of the form (2.1) are referred to as pure languages.

If in each production \( x \rightarrow y \) of \( P \) the left side \( x \) is a letter then we say that \( G \) is a pure context-free (briefly PCF) grammar. Languages generated by PCF grammars are referred as PCF languages.

If each production \( x \rightarrow y \) of \( P \) satisfies \( |x| \leq |y| \), then \( G \) is termed length-increasing. The abbreviation \( \text{PLI} \) is used when discussing pure length-increasing grammars and the languages generated by them.

Remark 1. Following Gabrielian (1970) and Buttelmann et al. (1974), we allow the set \( S \) of starting words or axioms to consist of a finite number of words, rather than of only one word. This gives us more flexibility. On the other hand, the problem of reducing the number of axioms will be discussed below.

Remark 2. Languages of sentential forms of context-free grammars (referred to as SF languages in Salomaa (1973b) and Harju and Penttonen (1978)) are a special case of PCF languages. An SF language is generated by a PCF grammar, where the set \( S \) consists of one word of length one.

Remark 3. Any context-free (resp. context-sensitive, type 0) language is obtained from a PCF (resp. PLI, pure) language \( L \) by intersecting \( L \) with the set of words \( \Sigma_1^* \) over some "terminal" alphabet \( \Sigma_1 \). It follows from the closure properties of the families in the Chomsky hierarchy that, conversely, all languages obtained in this fashion are context-free (resp. context-sensitive, type 0).

Remark 4. One can introduce also the notion of a pure context-sensitive grammar: all productions are of the form
\[
x_1 a x_2 \rightarrow x_1 z x_2, \quad a \in \Sigma \quad \text{and} \quad z \neq \lambda,
\]
i.e., \( a \) can be rewritten as \( z \) within the context \((x_1, x_2)\). Clearly, every language
generated by a pure context-sensitive grammar is PLI. Contrary to the case of ordinary phrase structure grammars, the converse does not hold true. For instance, the language generated by the PLI grammar

\[ G = ([a, b], \{a^2 \rightarrow b^4, b^2 \rightarrow a^4\}, \{a^2\}) \]

is not pure context-sensitive. The ordinary simulation of length-increasing productions by context-sensitive ones does not work here because one is not able to "hide" the intermediate words needed for the simulation: everything is in the language. This is also an example showing that ordinary reduction techniques do not, in general, work for pure grammars.

We consider next some examples. From now on we define pure grammars simply by listing the axioms and productions.

The language \( \{a^n b^n | n \geq 1\} \) is generated by the PCF grammar with the axiom \( acb \) and the only production \( c \rightarrow acb \).

If we omit the center marker \( c \) and consider the language \( L = \{a^n b^n | n \geq 1\} \), we observe first that \( L \) is generated by the PLI grammar with the axiom \( ab \) and the only production \( ab \rightarrow a^2 b^2 \). By analyzing the possible productions for \( a \) and \( b \) it is easy to see that \( L \) is not PCF. A detailed argument can be found in Buttelmann et al. (1974).

It is also shown in Buttelmann et al. (1974) that neither of the languages

\[ \{a^n b^n c^n | n \geq 1\} \quad \text{and} \quad \{a^n b^n | n \geq 1\} \cup \{a^n b^{2n} | n \geq 1\} \quad (2.2) \]

is pure. We shall discuss in the sequel several techniques for showing that a given language is not pure.

Following the customary terminology, we call two pure grammars equivalent if they generate the same language.

We conclude this section with some notions dealing with the mode of generation according to a pure grammar.

A pure grammar \( G \) is monogenic if, whenever \( w \) is in \( L(G) \) and \( w \rightarrow w' \), then there are unique words \( w_1 \) and \( w_2 \) such that

\[ w = w_1 x w_2, \quad w' = w_1 y w_2, \quad \text{and} \quad x \rightarrow y \text{ is a production} \]

and moreover, there is no word \( w'' \neq w \) such that \( w \rightarrow w'' \).

Thus, in a derivation according to a monogenic grammar, each word is uniquely determined by its predecessor, both regarding the production applied and the position of application. Each axiom determines a unique sequence of words. We shall see in Section 8 that monogenic PCF languages are rather restricted in nature and that there are PCF languages which can be generated by a monogenic PLI grammar but not by a monogenic PCF grammar.
A PCF grammar is deterministic if there is at most one production for each letter.

Assuming that a monogenic PCF grammar $G$ is reduced, i.e., every letter of the alphabet occurs in some word of $L(G)$, it is clear that $G$ is also deterministic. On the other hand, deterministic PCF grammars are not, in general, monogenic.

Consider a derivation $D$ according to a PCF grammar $G$ (i.e., a finite sequence of words, where the first word is an axiom and each word yields directly the next one). We say that $D$ is leftmost if the following condition is satisfied. Whenever $w$ and $w'$ are two consecutive words in $D$ and $w'$ results from $w$ by an application of the production $a \rightarrow y$, i.e.

$$w = w_1aw_2 \quad \text{and} \quad w' = w_1yw_2,$$

then no word in $D$ coming after $w'$ is obtained by applying a production to a letter in $w_1$.

It is clear that each word in the language generated by a PCF grammar can be generated by a leftmost derivation. We say that a PCF grammar $G$ is unambiguous if no word in $L(G)$ possesses two distinct leftmost derivations. Otherwise, $G$ is ambiguous. A PCF language $L$ is unambiguous if it is generated by an unambiguous PCF grammar. Otherwise, $L$ is (inherently) ambiguous.

We consider in this paper only the notion of ambiguity introduced above and based on leftmost derivations. A more general notion of ambiguity could be defined by calling $G$ ambiguous if some word in $L(G)$ possesses two distinct derivation trees. (It is then clear that if $G$ is ambiguous in the leftmost sense introduced above, then it is also ambiguous in this more general sense, but not necessarily conversely.) Under this more general notion of ambiguity every infinite PCF language over a one-letter alphabet is inherently ambiguous.

3. INTERRELATIONS TO THE CHOMSKY HIERARCHY

Our first theorem is essentially due to Büchi (1964). A more detailed exposition can be found in Salomaa (1969). The theorem is also contained in Buttelfmann et al. (1974) and Gabrielian (1970). We present the proof here for the sake of completeness and because similar techniques are quite widely applicable in the study of pure grammars.

**Theorem 3.1.** Every regular language is generated by a PLI grammar.

**Proof.** Let $R$ be a language over $\Sigma$ accepted by a finite deterministic automaton $A$ with the state set $Q$ and transition function $\delta$. Each word $w$ over $\Sigma$ induces a mapping $f_w$ of $Q$ into itself, defined by

$$f_w(q) = \delta(q, w) \quad \text{for each} \ q \in Q.$$
(Here $\delta(q, w)$ is the state into which the word $w$ takes the automaton from the state $q$.) There are only finitely many words $w$ inducing different mappings $f_w$. Let $m$ be the smallest integer such that every mapping induced by a word of length $m$ is already induced by some shorter word.

Consider now the PLI grammar

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

defined as follows. The set $S$ consists of all words in $R$ shorter than $m$. For each word $y$ of length $m$, choose a shorter word $x_y$ inducing the same mapping as $y$. Then the set $P$ consists of all productions

$$x_y \rightarrow y. \quad (3.1)$$

We shall now prove that

$$R = L(G). \quad (3.2)$$

To prove that the right side is contained in the left side, it suffices to show that, whenever (3.1) is applied to a word $z_1x_zy_2$ in $R$, then the resulting word $z_1y_2$ is also in $R$. But this is obvious: in whatever state $q$ the automaton $A$ is after reading $z_1$, both $x_y$ and $y$ map $q$ to the same state.

To prove that the left side of (3.2) is contained in the right side, we observe first that all words in $R$ shorter than $m$ are in $L(G)$. Proceeding inductively, we assume that all words in $R$ shorter than $p \geq m$ are in $L(G)$. Let $w$ be a word of length $p$ in $R$. (If there are no words of length $p$ in $R$, we have completed the inductive step.) Decompose

$$w = yz, \quad |y| = m.$$ 

Then the word $w' = x_yz$ is in $R$ and, by our inductive hypothesis, also in $L(G)$. Applying now the production $x_y \rightarrow y$ to $w'$, we see that $w$ is in $L(G)$, which completes the induction. $lacksquare$

The following two theorems are due to Gabrielian (1970).

THEOREM 3.2. The family of pure languages over a one-letter alphabet $\{a\}$ coincides with the family of regular languages over $\{a\}$.

Proof. The previous theorem implies that every regular language over $\{a\}$ is pure. The converse is essentially due to the fact that in the one-letter case context does not give any information provided we are dealing with sufficiently long words (longer than the longest left-hand side among the productions). More formally, the converse follows, for instance, by Theorem 9.1 below. $lacksquare$

THEOREM 3.3. There are nonrecursive pure languages.
Proof. Consider an arbitrary language $L$ which is undecidable and generated by some type 0 grammar $G$ with terminal alphabet $\Sigma$. Let $L'$ be the sentential form language of $G$. $L'$ is a pure language and $L \subseteq L'$ holds. Clearly, $L'$ cannot be decidable, since for any word $x \in \Sigma^*$ we have $x \in L$ iff $x \in L'$. The results obtained so far enable us to show that in each of the differences of two language families in the Chomsky hierarchy there is a pure language, as well as a language which is not pure. More specifically, we consider the differences

$$L(\text{RE}) - L(\text{REC}), L(\text{REC}) - L(\text{CS}), L(\text{CS}) - L(\text{CF}), L(\text{CF}) - L(\text{REG}), \tag{3.3}$$

where $L(\text{RE}), L(\text{REC}), L(\text{CS}), L(\text{CF}), L(\text{REG})$ denote the families of recursively enumerable, recursive, context-sensitive, context-free and regular languages, respectively.

**Theorem 3.4.** Each of the differences (3.3) contains both pure and nonpure languages.

Proof. That each of the differences (3.3) contains a pure language follows exactly as Theorem 3.3: each of the differences contains a phrase structure language $L$ generated by a phrase structure grammar $G$ with terminal alphabet $\Sigma$. Let $L'$ be the sentential form language of $G$. Clearly, $L'$ is still in the larger language family. Since $L = L' \cap \Sigma^*$ and each of the families in the Chomsky hierarchy is closed under intersection with regular languages, $L'$ is not in the smaller family of languages.

The second of the languages (2.2) is an example of a nonpure language in the difference $L(\text{CF}) - L(\text{REG})$. Examples of nonpure languages in the other differences (3.3) are obtained by Theorem 3.2: there are languages over \{a\} in each of these differences. \[ \]

We conclude this section with a result which enables us to construct in a simple way languages that are not PCF. By this result languages such as $a^i b^i (a b)^*$ or \{a^i (a b)^i (a, b)^i \mid i \geq 1\} are not PCF languages.

**Lemma 3.5.** Assume that $L \subseteq \{a, b\}^*$ satisfies the following two conditions:

(i) every word in $L$ contains $a^3$ and $b^3$ as subwords,

(ii) there exists a constant $c > 0$ such that each word $x$ of $L$ of length $n$ contains a subword $y$ of length $cn$ and $y$ does not contain two consecutive occurrences of the same letter.

Then $L$ is not PCF.

Proof. (By contradiction). Suppose $L = L(G)$, $G$ a PCF. Note that $G$ cannot contain a production of the form $a \rightarrow b^i$ ($i \geq 0$) (since by applying such a production to some word $x$ of $L$ a word consisting solely of $b$'s can be
obtained). Similarly, productions \( b \rightarrow a^i \ (i \geq 0) \) are impossible. Note next that productions \( a \rightarrow z \) with \( |z| \geq 2 \) and \( z \) a subword of \( ababab \cdots ab \) are impossible (since they would allow us to generate words not containing \( a^3 \)). Similarly, productions \( b \rightarrow z \), \( z \) as above, are impossible. Hence all nontrivial productions of \( G \) (trivial productions are \( a \rightarrow a, b \rightarrow b \)) are of the form 
\[ \alpha \rightarrow u\beta v, \]
where \( \alpha, \beta \in \{a, b\} \) (i.e. right side contains two consecutive \( a \)'s or \( b \)'s) and \( u, v \in \Sigma^* \). Let \( q \) be the length of longest right side of any production. Choose some \( n \) such that \( cn > 2q \). Choose an arbitrary word \( x \) with \( |x| \geq n \) in \( L \) and apply to each symbol of \( x \) a nontrivial production. In the word \( y \) obtained, two groups of two consecutive symbols are at most \( 2q - 2 \) positions apart. Hence each subword of length \( cn > 2q \) contains two consecutive occurrences of the same letter. This contradiction completes the proof.

### 4. PCF Languages over \{a\}

This section is devoted to a detailed study of PCF languages over a one-letter alphabet. By Theorem 3.2 or by the fact that every context-free language over \{a\} is regular, we conclude first that all PCF languages over \{a\} are regular. On the other hand, all regular languages over \{a\} are not PCF. For instance, the language
\[ \{a^2\} \cup \{a^{2n+1} \mid n \geq 0\} \quad (4.1) \]
is not PCF. We shall present in this section an easily decidable characterization of PCF languages over \{a\}. Moreover, we obtain reduction results concerning the PCF grammars involved. For instance, we present a method of minimizing the number of axioms.

Let \( R \) be a regular language over \{a\}. Consider the minimal state deterministic finite automaton accepting \( R \). It is easy to see that \( R \) can be written as a disjoint union
\[ R = R_F \cup R_p, \quad \text{where} \quad (4.2) \]

(i) \( R_F \) is a finite language (the "initial mess") such that every word in \( R_F \) is shorter than the shortest word in \( R_p \),

(ii) \( R_p \) (the "periodic part") is of the form
\[ (a^{m_1} \cup \cdots \cup a^{m_k})a^p)^k \quad k \geq 0, \quad p > 0, \quad (4.3) \]

(iii) the period \( p \) is the shortest possible, i.e. there is no representation \((4.2)\) where the regular expression \((4.3)\) for \( R_p \) has a smaller value of \( p \), and

(iv) no two of the numbers \( m_i \) are congruent modulo \( p \).
If $R$ is finite, then in (4.3) we choose $k = 0$. (The same effect could be obtained by allowing $p = 0$.) In fact, it was shown in Salomaa (1964) that the representation (4.2) is unique, provided the borderline between the initial mess and the periodic part is defined in a suitable fashion. For our purposes it is sufficient to assume that condition (iii) is satisfied: $p$ is minimal. Throughout this section, we assume this to be the case, and that condition (iv) is satisfied. (In fact, if some of the $m_i$'s were congruent modulo $p$, the larger one could be removed without affecting the language.) The numbers $m_1, \ldots, m_k$ determine $k$ residue classes modulo $p$. Observe that if $p > 1$ then $k < p$.

For instance, the language (4.1) has $p = 2$ and $k = 1$. The language

$$\{a^{5n} \mid n \geq 1\} \{a^{7n} \mid n \geq 1\}$$

has $p = k = 1$. The reader might want to determine $R_v$ for the language (4.4).

The following lemma is the basic tool for considering PCF languages over $\{a\}$.

**Lemma 4.1.** Assume that a regular language $R$ over $\{a\}$ is generated by a reduced PCF grammar $G$. Then every production in $G$ is of the form

$$a \rightarrow a^{n+1}, \quad n \geq 0.$$  

**Proof.** We may assume that $R$ is infinite. Suppose the lemma is not true: $G$ has a production

$$a \rightarrow a^{n+p+1}, \quad 0 < q < p.$$  

(4.5)

We now apply the production (4.5) to the words in the periodic part $R_p$. Applying (4.5) to the words of the residue class determined by $m_i$ ($1 \leq i \leq k$), we see that $m_i + q$ also determines a residue class. Applying then (4.5) to the words of the residue class determined by $m_i + q$, we see that $m_i + 2q$ also determines a residue class. In general, we infer that, for any $t \geq 0$, $1 \leq i \leq k$, $m_i + tq$ determines a residue class. But this implies that $p$ can be replaced by a smaller number, which contradicts the minimality.

We are now in the position to give a characterization of PCF languages over $\{a\}$.

**Theorem 4.2.** A regular language $R$ over $\{a\}$ is PCF if and only if either $R$ is finite or else in the representation (4.2) every word of $R_v$ belongs to one of the residue classes of $R$ (i.e., to one of the residue classes determined by the numbers $m_1, \ldots, m_k$ in (4.3)).

**Proof.** Consider the "if"-part. Clearly, every finite language is generated by a PCF grammar, where the set of productions is empty. Assume that $R$ is infinite and that every word in the "initial mess" belongs to one of the residue
classes (but within each residue class there may be arbitrary gaps before the beginning of the periodic part). We now choose the smallest integer $t$ such that the production

$$a \rightarrow a^{tp+1}$$

is compatible with the language $R$ in the following sense: whenever (4.6) is applied to a word in $R$, the result of the application is also in $R$. Such an integer $t$ exists: it suffices to make sure that an application of (4.6) always leads from the "initial mess" to the periodic part. The compatibility of all productions (4.6) with a sufficiently large $t$ follows by our assumption concerning the words in $R_F$.

We now construct a PCF grammar $G$ as follows. The only production is (4.6). The axioms are the words in $R_F$ and, in addition, the words from $R_F$ belonging to the language (cf. (4.3)).

$$(a^m \cup \cdots \cup a^{m_k})(1 \cup a^p \cup a^{2p} \cup \cdots \cup a^{(t-1)p}).$$

It is easy to see that $L(G) = R$.

Consider the "only if"-part. Assume that $R$ is infinite and that $R_F$ contains a word $w$ not belonging to any of the residue classes of $R$. It is now an immediate consequence of Lemma 4.1 that $R$ cannot be PCF.

The following theorem is an immediate corollary of Theorem 4.2 and its proof.

**Theorem 4.3.** PCF languages over \{a\} constitute a proper subfamily of regular languages over \{a\}, and it is decidable whether or not a given regular language over \{a\} is PCF. All cofinite languages over \{a\} are PCF. Every PCF language over \{a\} is generated by a PCF grammar with at most one production.

Theorem 4.3 shows that one production is always sufficient. We now turn to the question of minimizing the number of axioms. For $i = 1, 2, \ldots$, denote by

$$L_i(\text{PCF})$$

The family of PCF languages over \{a\}, generated by PCF grammars with at most $i$ axioms. It follows by Lemma 4.1 that no language with more than $i$ residue classes belongs to $L_i(\text{PCF})$. On the other hand, if $R$ has at most $i$ residue classes, it does not necessarily belong to $L_i(\text{PCF})$. For instance, the language

$$a^2 \cup a^3(a^2)^*$$

has just one residue class but cannot be generated with fewer than two axioms ($a^2$ and $a^3$).
To find out the minimal number of axioms, we proceed as follows. Let \( R \) be a regular PCF language over \( \{a\} \), with the period \( p \). (We assume that \( R \) is infinite. If \( R \) is finite, the minimal number of axioms equals the cardinality of \( R \).) We find the integer \( t \) as in the proof of Theorem 4.2 and test which among the productions

\[
a \rightarrow a^{i+p+1}, \quad 1 \leq i \leq 2t - 1,
\]

are compatible with \( R \). Let \( P \) consists of those productions (4.8) which were found to be compatible with \( R \).

Clearly, if \( G \) is any reduced PCF grammar for \( R \) then all productions in \( G \) must be compatible with \( R \). It is also obvious that every compatible production

\[
a \rightarrow a^{i+p+1}, \quad i \geq 2t,
\]

can be simulated by the productions in \( P \). In this sense, \( P \) is the maximal set of compatible productions.

We now consider an enumeration of \( R \) according to the word length. Starting with the shortest word, we test of each word \( w \) whether it is generated from the shorter words by the productions in \( P \). If this is not the case, \( w \) must be an axiom. It is clear that the procedure must terminate: once we are in the periodic part and have found out that all words within one cycle have been generated from words also in the periodic part, we may stop. The number of words \( w \) listed as axioms during this procedure gives the minimal number of axioms. Thus, we have established the following results.

**Theorem 4.4.** The families (4.7) constitute, for \( i = 1, 2, \ldots \), an infinite hierarchy of strictly increasing language families. Given a PCF language \( L \) over \( \{a\} \), we may effectively determine the smallest \( i \) such that \( L \) is in \( L^i(\text{PCF}) \).

By Theorem 4.3, every PCF language over \( \{a\} \) can be generated by a PCF grammar \( G \) with only one production. If superfluous axioms are removed from \( G \), the resulting PCF grammar will be unambiguous. Hence, the following theorem holds true.

**Theorem 4.5.** Every PCF language over \( \{a\} \) is unambiguous.

**Theorem 4.6.** Every pure language \( L \) over \( \{a\} \) can be generated by a pure grammar with at most one production.

**Proof.** By Theorem 3.2, \( L \) is regular. Hence, the decomposition 4.2 can be given for \( L \). (We again assume that \( L \) is infinite, the finite case being trivial.) Observe now that the productions

\[
a^i \rightarrow a^{i+n} \quad \text{and} \quad a^{i+j} \rightarrow a^{i+j+n}
\]
have exactly the same effect when applied to words of length at least \( i + j \), whereas the latter cannot be applied to shorter words. Thus, it suffices to choose sufficiently many axioms and a production \( a^i \rightarrow a^{i+p} \), where \( p \) is the period of \( L \) and \( i \) exceeds the length of words in the “initial mess.”

Languages over \( \{a\} \) are a special case of bounded languages. The characterization of bounded PCF languages is essentially more involved, because of the simple reason that such languages need not be regular. PCF sublanguages of \( a^*b^* \), where \( a \) and \( b \) are letters, are always regular and it is easy to characterize such languages. This characterization can be extended to concern PCF sublanguages of \( a_1^*a_2^* \cdots a_n^* \), where the \( a_i \)'s are distinct letters. The latter case is, however, considerably more difficult and contains also nonregular languages.

We conclude this section with some results concerning PCF sublanguages of \( w^* \), where \( w \) is a nonempty word over an alphabet \( \Sigma \). These languages are very closely related to languages over \( \{a\} \). Given a subset \( L \) of \( w^* \), we define in the natural way the corresponding language \( L_a \) over \( \{a\} \): \( a^i \) belongs to \( L_a \) if and only if \( w^i \) belongs to \( L \).

**Theorem 4.7.** A language \( L \subseteq w^* \) where \( w \) is of the form \( w = dz \), where \( d \) is a letter not occurring in \( z \), is PCF if and only if the corresponding language \( L_a \) is PCF. All of the following conditions (i)–(iv) are equivalent, for arbitrary \( w \).

(i) \( L \subseteq w^* \) is pure.

(ii) The corresponding language \( L_a \) is pure.

(iii) \( L \subseteq w^* \) is regular.

(iv) \( L \subseteq w^* \) is context-free.

Every pure language \( L \subseteq w^* \) is generated by a PLI grammar, where each production is of the form \( w^i \Rightarrow w^j \).

**Proof.** If \( L \) is PCF then clearly \( L_a \) is PCF. The productions \( a \rightarrow a^i \) are obtained from derivation steps \( w \Rightarrow w^i \) for \( L \).

Assume that \( L_a \) is PCF. \( L \) is now generated by the PCF grammar \( G \), obtained from the PCF grammar \( G_a \) for \( L_a \) in the following fashion. The axioms of \( G \) are obtained by applying the homomorphism

\[
h(a) = w
\]

to the axioms of \( G_a \). Each production \( a \rightarrow aa^i \) in \( G_a \) becomes the production \( d \rightarrow d(zd)^i \) in \( G \). This proves the first sentence.

The equivalence of the conditions (ii)–(iv) is a consequence of Theorem 3.2. That (ii) implies (i) is seen in the same way as the corresponding statement concerning PCF grammars. The converse implication is established by considering all derivations \( w^i \Rightarrow w^j \) according to the pure grammar \( G \) for \( L \), where
5. LENGTH SETS

Many of the phenomena typical for pure grammars and languages can be seen already from the generated length sets. The size of the gaps in the length set of a pure language is always bounded by a constant. This follows because all intermediate words in a derivation belong to the language and, thus, the size of the gaps is bounded by the greatest difference between the lengths of the right and left side of a production. Consequently, sets such as

\[ \{n^2 \mid n \geq 1\} \quad \text{and} \quad \{2^n \mid n \geq 1\} \]

are not length sets of pure languages.

We begin with a characterization of length sets of PCF languages. The characterization makes use of length sets of PCF languages over \{a\}. However, the latter length sets possess a particularly simple characterization due to Theorem 4.2.

**Theorem 5.1.** All of the following conditions (i)–(iii) are equivalent for a set \(S\) of nonnegative integers.

(i) \(S\) is the length set of a PCF language.

(ii) \(S\) is the length set of a context-free language.

(iii) \(S = F \cup S_1\), where \(F\) is a finite set and \(S_1\) is the length set of a PCF language over \{a\}.

**Proof.** Observe that the family of length sets of context-free languages coincides with the family of length sets of regular languages over \{a\}. By Theorem 4.2, the latter family equals the family of sets representable in the form (iii). On the other hand, every set representable in the form (iii) is the length set of a PCF language over \(\{a, b\}\): there are no productions for \(b\), and \(b^i\) is an axiom if and only if \(i\) is in \(F\). 

Theorem 5.1 shows that the length sets of context-free and PCF languages coincide. The result does not carry over to the context-sensitive case because of the property of bounded gaps, mentioned at the beginning of this section. Thus, the family of length sets of PLI languages is properly contained in the family of length sets of context-sensitive languages.

The customary reduction or normal form theorems do not extend to pure (resp. PCF, PLI) grammars. This follows by arguments based on length sets.
alone. For instance, the language families $K_i$, $i = 1, 2, \ldots$, generated by PLI grammars $G$ such that the difference

$$|y| - |x|$$

has the upper bound $i$, for all productions $x \rightarrow y$ of $G$, constitute an infinite strictly increasing hierarchy.

From the point of view of length sets, we may assume that the left side of each production is of length $\leq 2$. More specifically, for each PLI grammar $G$, a PLI grammar $G'$ with this property and generating the same length set as $G$ can be constructed. However, from the point of view of languages, not even this reduction result is valid. For instance, consider the PLI grammar $G$ with the axiom $LMa^3R$ and productions

$$Ma^3 \rightarrow b^6M, \quad MR \rightarrow ZR$$
$$b^3Z \rightarrow Za^6, \quad LZ \rightarrow LM.$$  

There is no PLI grammar $G'$ equivalent to $G$ such that the lengths of the left-hand sides of the productions in $G'$ are bounded by 2, as is easily seen by contradiction. For suppose such a PLI grammar $G'$ did exist. Let $m$ be the largest difference between the length of right hand side and left hand side of any production of $G$. Choose $i \geq m$ such that the word $y = Lb^6iMR$ is not an axiom of $G'$. Consider the word $x$ preceding $y$ in the derivation of $y$, i.e. $x \Rightarrow Lb^6iMR$. Since $|Lb^6iMR| - |x| \leq m$ must hold, $x$ must be of the form $x = Lb^{6(j-1)}Ma^3jR$ for some $j$, $1 \leq j \leq m/3$. Thus a production $uMa^3j \rightarrow ub^6M$ must occur in $G'$ for some word $u$. This contradicts the assumption that the length of the left hand side of each production does not exceed 2.

Although the size of the gaps is bounded, one may construct examples of PLI length sets, where for instance, the difference between two consecutive word lengths equals 1 or 2 but where 2 does not occur at regular intervals. One such example is obtained by an easy modification of the monogenic PLI grammar presented in Section 8: in connection with each successful addition to the binary number we also add a dummy letter to the word. Then the word length increases by 2 if and only if the length of the binary number increases. This happens at longer intervals when the binary number becomes longer.

Because of properties of length sets alone, many of the well-known language families lie mostly outside the family of pure languages. For instance, an infinite D0L language can be pure only in case it is of linear growth. On the other hand, not every D0L language of linear growth is pure.

$L = \{a^nf b^nf c^n \mid n \geq 1\}$ is such an example. $L$ is clearly a linear growth D0L language. To see that $L$ is not pure, suppose $L = L(G)$ where $G$ is a pure grammar. Choose $m$ larger than the length of any left- or right hand side of any production of $G$. Consider some $n > m$ such that $y = a^nf b^nf c^n$ is not
an axiom of $G$. Let $x$ be the word preceding $y$ in the derivation of $y$. Clearly, $x \neq y$ may be assumed. Thus $x \Rightarrow a^n b^n c^n$. Hence $x = x_1 u x_2$, $a^n b^n c^n = x_1 e x_2$, $u \rightarrow v$ is a production of $G$ and $|v| < n$. Thus $x$ is not in $L$, a contradiction.

Clearly, all PLI languages are context-sensitive. Thus, by Theorem 3.3, PLI languages constitute a proper subfamily of pure languages. Our next theorem shows that this difference exists already at the level of length sets.

**Theorem 5.2.** There is a pure grammar $G$ such that no PLI grammar generates the length set $LS(L(G))$.

**Proof.** Consider a type 0 language $L \subseteq a(a^2)^*$ which is not context-sensitive. We generate $L$ by a type 0 grammar $G$ working in such a manner that all sentential forms, with the exception of the final ones, are of even length. We now consider $G$ as a pure grammar. Then the length set $LS(L(G))$ cannot be generated by a context-sensitive grammar (let alone PLI) because, otherwise, the language

$$L(G) \cap a(a^2)^* = L$$

would be context-sensitive, which is a contradiction.

6. PROBLEMS DEALING WITH REGULAR LANGUAGES

We consider in this section various problems about the interrelation between PCF and regular languages. As we already have seen, these two language families are incomparable.

We show first that under certain circumstances the language generated by a PCF grammar is always regular. The starting point is the left-linear form of productions. This alone is not sufficient: even if all productions are of the form $a \rightarrow ax$, where $x$ does not contain the letter $a$, we might still get self-embedding situations (cf. Hagauer (1978)). However, the following result is valid.

**Theorem 6.1.** Assume that a PCF grammar $G$ satisfies the following condition. For each letter $a$, whenever $a \Rightarrow x ay$ is a derivation then $x = \lambda$. Then $L(G)$ is regular.

**Proof.** For the sake of this proof only, call PCF grammars satisfying the above condition leftbound. Further, for a PCF grammar $G = (\Sigma, P, S)$ call a symbol $a \in \Sigma$ productive if $P$ contains some production with left side $a$.

We prove Theorem 6.1 by establishing the following result.
If \( G \) is a leftbound PCF grammar with \( n > 1 \) productive symbols, then we can construct a leftbound PCF grammar \( G' \) with \( m < n \) productive symbols such that \( L(G) = \tau(L(G')) \), where \( \tau \) is a regular substitution. This proves our theorem because clearly a leftbound PCF grammar with \( \leq 1 \) productive symbols generates a regular language.

Let \( G = (\Sigma, P, S) \) be an arbitrary leftbound PCF grammar. Let \( \Delta \subseteq \Sigma \) be the set of all productive symbols of \( G \). We assume \( \Delta \neq \emptyset \). To obtain the PCF grammar \( G' \) described we roughly proceed as follows:

We first define an equivalence relation on \( \Delta \); we then define a set of “lowest” equivalence classes; we finally choose one of those lowest equivalence classes and establish that deleting all productions for symbols of such an equivalence class gives rise to a leftbound PCF grammar \( G' \) as described.

We now turn to the technical details. We call two symbols \( a, b \) of \( \Delta \) equivalent and write \( a \sim b \) iff \( a \Rightarrow bx \) and \( b \Rightarrow ay \) holds for some \( x, y \in \Sigma^* \). Observe that \( \sim \) is an equivalence relation and hence \( \Delta \) the disjoint union of a finite number of equivalence classes \( C_1, C_2, \ldots, C_t \).

To be able to define the notion of “lowest equivalence classes” we first define a partial ordering “to dominate” among the equivalence classes, and then a partial ordering “to be below” among all those equivalence classes not dominating any others.

More precisely, for equivalence classes \( C_i, C_j \) we say \( C_i \) dominates \( C_j \) and write \( C_i > C_j \) iff \( a \Rightarrow bx \) holds for some \( a \in C_i, b \in C_j \) and \( x \in \Sigma^* \). Observe that \( C_i > C_j \) indeed implies that for every \( a \in C_i \) and every \( b \in C_j \) there exists some \( x_{a,b} \) in \( \Sigma^* \) such that \( a \Rightarrow bx_{a,b} \). Observe further that for \( i \neq j \) it is impossible to have both \( C_i > C_j \) and \( C_j > C_i \); \( C_i > C_j \) implies \( a \Rightarrow bx \) and \( C_j > C_i \) implies \( b \Rightarrow ay \) (for suitable \( x, y \)), hence \( a \sim b \), hence \( C_i = C_j \), a contradiction. Thus, between any two equivalence classes \( C_i, C_j \) at most one of the relations \( C_i > C_j \) or \( C_j > C_i \) holds. Since \( > \) is clearly transitive, \( > \) is a partial ordering.

Consider now the set \( U \) of all equivalence classes which do not dominate any others, \( U = \{ C_i \mid C_i > C_j \text{ does not hold for any } j \} \).

For equivalence classes \( C_i, C_j \) of \( U \) we call \( C_i \) below \( C_j \) and write \( C_i \prec C_j \) if \( a \Rightarrow bx \) holds for some \( a \in C_i, b \in \Delta \) (indeed \( b \in C_j \)), \( c \in C_j \) and \( x, y \in \Sigma^* \).

We show that \( \prec \) is a partial ordering on \( U \). Observe first that for different \( C_i, C_j \) of \( U \) it is impossible to have both \( C_i \succ C_j \) and \( C_j \succ C_i \); \( C_i \succ C_j \) implies \( a \Rightarrow bx \) for some \( a \in C_i, c \in C_j ; C_j \succ C_i \) implies \( a' \Rightarrow b'x'c'y' \) for some \( a' \in C_j, c' \in C_i \) \((x, y, x', y' \text{ some words of } \Sigma^*) \). Since \( c \) and \( a' \) are both in \( C_j \) and \( a \) and \( c' \) are both in \( C_i \) we have \( c \Rightarrow a'z \) and \( c' \Rightarrow aw \) for suitable \( z \) and \( w \), i.e. \( a \Rightarrow bx \Rightarrow bx'c'y' \Rightarrow bxb'x'c'y'z \Rightarrow bxb'x'aw \). Thus we have \( a \Rightarrow uaw \) with \( u \neq \lambda \), a contradiction.

That \( \succ \) is transitive is equally evident: Suppose \( C_i \succ C_j \) and \( C_j \succ C_k \) hold, i.e. \( a \Rightarrow bx \) for \( a \in C_i, c \in C_j \) and \( c \Rightarrow dx \) for \( d \in C_j, e \in C_k \) and \( x, y \in \Sigma^* \). Then \( a \Rightarrow bx \Rightarrow bx'de \Rightarrow bx'dx'e \), i.e. \( C_i \succ C_k \).
Any equivalence class of $U$ which has no equivalence class below it is called a **lowest** equivalence class.

Choose one equivalence class $D$ containing at least one productive symbol and as low as possible. Note that for every symbol $a \in D$ each production is of the form $a \rightarrow bx$ or $a \rightarrow x$ with $b \in D$ and $x$ containing no productive symbol, i.e., $x \in (\Sigma - D)^*$. Define the PCF grammars:

\[
G_a = (\Sigma, P, \{a\}) \text{ for every } a \in D \text{ and } \\
G' = (\Sigma, P', S), \text{ where } P' \text{ is } P \text{ with all productions for symbols of } D \text{ deleted.}
\]

Clearly, $G'$ contains fewer productive symbols than $G$ and $L(G) = \tau(L(G'))$, where

\[
\tau(a) = \begin{cases} 
  a & \text{for } a \in \Sigma - D \\
  L(G_a) & \text{for } a \in D.
\end{cases}
\]

It remains to show that $L(G_a)$ is regular for every $a \in D$. We do this by constructing a left linear grammar $H_a$ generating $L(G_a)$.

Define a set $\Phi$ of nonterminals, $\Phi = \{N_a \mid a \in D\}$. Consider the left linear grammar $H_a = (V, \Sigma, P_a, N_a)$ with $V = \Phi \cup \Sigma$ and

\[
P_a = \{N_a \rightarrow a \mid a \in D\} \\
\cup \{N_a \rightarrow N_bx \mid a, b \in D, a \rightarrow bx \in P\} \\
\cup \{N_a \rightarrow x \mid a \in D, a \rightarrow x \in P\}.
\]

Clearly, $L(H_a) = L(G_a)$, concluding the proof.

Clearly, the symmetric right-linear version of Theorem 6.1 can be established in the same way. It seems likely that regularity can be established also in certain self-embedding situations. We hope to be able to return to this matter in a future paper.

We now turn to the discussion of some decision problems. The following result, apart from being of interest on its own right, is quite useful in many decision problems. As in Section 4, we say that a production $a \rightarrow b$ with a language $L$ if, whenever $a$ is applied to a word in $L$, then the result of application is also in $L$.

**Theorem 6.2.** It is decidable, whether or not a given production $x \rightarrow y$ (not necessarily context-free) is compatible with a given regular language $R$.

**Proof.** Consider the minimal deterministic finite automaton $A$ accepting $R$. Let the state set and transition function of $A$ be $S$ and $\delta$, respectively. For any $s$ in $S$, denote by $L(s)$ the language accepted by the automaton obtained from $A$ by letting $s$ be the initial state. To prove Theorem 6.2, we establish
the following stronger statement. The production \( x \rightarrow y \) is compatible with \( R \) if and only if

\[
L(\delta(s, x)) \subseteq L(\delta(s, y)), \quad \text{for every } s \text{ in } S. \tag{6.1}
\]

Assume first that \( x \rightarrow y \) is compatible with \( R \). Consider an arbitrary but fixed state \( s \) and choose a word \( z \) with the property

\[
\delta(s_0, z) = s,
\]

where \( s_0 \) is the initial state of \( A \). Consider a word \( w \) belonging to \( L(\delta(s, x)) \). Hence, \( zxw \) is in \( R \). Because \( x \rightarrow y \) is compatible with \( R \), we infer that \( zyw \) is in \( R \). But this implies that \( w \) is in \( L(\delta(s, y)) \). Hence, (6.1) is satisfied.

Assume, secondly, that (6.1) is satisfied. Consider a word \( zxw \) in \( R \). Denote \( s = \delta(s_0, z) \). Thus, \( w \) is in \( L(\delta(s, x)) \) and, by (6.1), also in \( L(\delta(s, y)) \). But this means that \( zyw \) is in \( R \) and, consequently, \( x \rightarrow y \) is compatible with \( R \). |
not generated by any PCF grammar bounded by \( f(\#(R)) \), where \( \#(R) \) denotes the number of states in the minimal finite deterministic automaton accepting \( R \). This follows because, otherwise, we could construct by Theorem 6.2 the maximal possible PCF grammar for \( R \) and then test the equivalence by Theorem 6.3. Thus, we could decide the PCF-ness of \( R \), which we assumed not to be the case. Similar conclusions can be made if the regularity of a PCF language is undecidable.

The following results are direct consequences of Theorem 6.3 (the decidability part).

**Theorem 6.4.** If a regular language \( R \) is known to be PCF, then a PCF grammar for \( R \) can be effectively constructed. If a PCF language \( L \) is known to be regular, then a finite automaton accepting \( L \) can be effectively constructed.

### 7. Decision Problems

We discuss in this section some further decision problems. By Salomaa (1973b), it is undecidable whether two given PCF grammars are equivalent, and also whether the intersection of two given PCF languages is empty or infinite. By Buttelmann et al. (1974), it is undecidable whether a given type 0 language is pure. Membership problem is decidable for PLI languages but, by Theorem 3.3, undecidable for pure languages.

The following two theorems present decidability results very typical for pure languages.

**Theorem 7.1.** For pure grammars \( G \) with the alphabet \( \{a, b\} \), it is decidable whether or not \( b \) occurs in some word in \( L(G) \). For PLI grammars \( G \) with the alphabet \( \{a, b, c\} \), it is undecidable whether or not \( c \) occurs in some word in \( L(G) \).

**Proof.** The first sentence follows by Theorem 3.2. If \( b \) does not occur in any of the axioms, we can decide (by Theorem 3.2) whether or not any of the productions having \( b \) in the right-hand side becomes applicable.

Consider the second sentence. We encode an arbitrary context-sensitive grammar \( G \) with \( n \) letters \( \alpha_i \) (terminals and nonterminals) to the alphabet \( \{a, b\} \) by the following homomorphism:

\[
h(\alpha_i) = a^i b^{n+1-i}, \quad 1 \leq i \leq n.
\]

\( h \) is extended in the natural way to concern productions. Denote by \( h(G) \) the resulting grammar. It is undecidable whether or not a specified letter, say \( \alpha_1 \), appears in some sentential form of \( G \). We now add to \( h(G) \) the production

\[
ab^n \rightarrow c^{n+1}
\]
and consider \( h(G) \) as a PLI grammar over the alphabet \( \{a, b, c\} \). Then \( c \) appears in some word in \( L(h(G)) \) if and only if \( \alpha_1 \) appears in some sentential form of \( G \).

**Theorem 7.2.** It is decidable whether or not a given PLI grammar \( G \) with the alphabet \( \Sigma \) generates the language \( \Sigma^* \).

**Proof.** Let \( k \) be the length of the longest axiom in \( G \). We test first whether or not all words of length \( \leq k + 1 \) are in \( L(G) \). If not, then clearly \( L(G) \neq \Sigma^* \). Otherwise, we claim that \( L(G) = \Sigma^* \). Indeed, consider an arbitrary word \( w \) of length \( k + 2 \). Write the decomposition

\[
w = w_1a, \quad |w_1| = k + 1, \quad a \in \Sigma.
\]

By the choice of \( k \) and because \( w_1 \in L(G) \), there is a word \( w_2 \) in \( L(G) \) such that

\[
w_2 \succeq w_1 \quad \text{and} \quad |w_2| \leq k.
\]

Hence,

\[
w_2a \succeq w_1a = w, \quad |w_2a| \leq k + 1.
\]

This implies that \( w \) is in \( L(G) \). The equation \( L(G) = \Sigma^* \) follows now by a straightforward induction.

The emptiness problem is trivial for pure languages. The infinity problem is more involved. It can be shown undecidable even for PLI grammars by the following argument.

**Theorem 7.3.** It is undecidable whether or not a given PLI language is infinite.

**Proof.** We apply a reduction to the Post Correspondence Problem in the following way. Let \( \text{PCP} \) be an arbitrary instance, determined by the lists of words \((a_1, ..., a_n)\) and \((b_1, ..., b_n)\). We now construct a PLI grammar \( G \) such that \( L(G) \) is infinite if and only if \( \text{PCP} \) has no solution. In fact, \( G \) will be monogenic. We only indicate how \( G \) works, the details of the construction are omitted.

The axiom of \( G \) is \( A1BCD \). \( G \) generates between the markers \( A \) and \( B \) all words over the alphabet \( \{1, ..., n\} \) until eventually a solution to \( \text{PCP} \) is found. It generates these words viewed as \( n \)-ary numbers, always adding 1 to the previous number. Every time a particular word \( i_1 \cdots i_s \) has been generated, \( G \) prints \( \alpha_{i_1} \cdots \alpha_{i_s} \) between \( B \) and \( C \) and \( \beta_{i_1} \cdots \beta_{i_s} \) between \( C \) and \( D \). After this \( G \) checks whether or not

\[
\alpha_{i_1} \cdots \alpha_{i_s} = \beta_{i_1} \cdots \beta_{i_s}.
\]
If so, the derivation terminates. Otherwise, new markers $C$ and $D$ are inserted next to $B$, leaving the previously produced material as garbage to the right end. After this, $G$ adds 1 to the word over $\{1,\ldots, n\}$ between $A$ and $B$. 

8. Ambiguity and Monogenicity

We first give just an example of an inherently ambiguous PCF language.

**Theorem 8.1.** The language $L = \{a^ib^j | i \geq j \geq 0\}$ is an inherently ambiguous PCF language.

**Proof.** $L$ is generated by the PCF grammar with the axiom $b$ and productions

$$b \rightarrow ab \quad \text{and} \quad b \rightarrow aba.$$ 

To see that $L$ is inherently ambiguous, we argue as follows. Consider an arbitrary PCF grammar $G$ for $L$. $G$ cannot have any productions for $a$ (apart from the identity $a \rightarrow a$), because no production for $a$ is compatible with $L$. This implies that $G$ must have at least two productions for $b$. They must occur in the same derivation. By interchanging their order, we obtain a different derivation. Hence, $G$ is ambiguous.

We call languages generated by monogenic PCF grammars *PCF-monogenic*. *PLI-monogenic* languages are defined analogously. It is clear that every reduced monogenic PCF grammar is deterministic. On the other hand, the language $\{a, b\}^+$ is not PCF-monogenic but it is generated by the deterministic PCF grammar with the axiom $b$ and productions

$$b \rightarrow a \quad \text{and} \quad a \rightarrow b^2.$$ 

The language generated by the PCF grammar with the axiom $c$ and productions

$$c \rightarrow aca \quad \text{and} \quad c \rightarrow bcb$$

is not generated by any deterministic PCF grammar.

The following characterization result follows readily from the definitions.

**Theorem 8.2.** Every PCF-monogenic language is a finite union of languages of the form

$$\{xu^nbv^n | n \geq 1\},$$

where $x, u, v, y$ are words and $b$ is a letter.
Proof. Let $G = (\Sigma, P, S)$ be a monogenic PCF grammar. Call a symbol $a$ in $\Sigma$ productive, if for some $z \neq a$ the production $a \rightarrow z$ occurs in $G$. Observe that no word of $G$ can contain two different productive symbols. From this it is easy to deduce that every axiom $w$ of $G$ defines a derivation

$w = x_0 \rightarrow x_1 \rightarrow \ldots \rightarrow x_n \rightarrow \ldots$

Let $i$ be the smallest integer with $b_i = b_j$ for some $j > i$. Then the words generated by $w$ are $x_0, x_1, \ldots, u_0 u_1 \ldots u_{i-1} b_i \ldots v_{i-1} v_i \ldots v_n$, proving the theorem.

The next theorem, our main result in this section, shows that monogenicity can be obtained in the transition from PCF to PLI grammars.

**Theorem 8.3.** There is a PCF language $L$ which is not PCF-monogenic but is PLI-monogenic.

**Proof.** Consider the PLI grammars $G$ with the axiom $B10AE$ and productions

$0A \rightarrow 1A,$

$1A \rightarrow C0,$

$i0C \rightarrow i1Z,$ for $i = 0, 1$

$i1C \rightarrow iC0,$ for $i = 0, 1$

$B1C \rightarrow B10Z,$

$Z00 \rightarrow 0Z0,$

$Z0E \rightarrow 0AE.$

Clearly, $G$ is monogenic because every word in $L(G)$ contains exactly one of the letters $A, C, Z$; and the left side of each production contains one of these letters in a unique combination. Starting with the number 10, $G$ generates all numbers in the binary notation.

We now claim that

$L(G) = \{B1xaE \mid x \in \{0, 1\}^+\}$

$\cup \{B1x0iE \mid x \in \{0, 1\}^{*}, i \geq 1\}$

$\cup \{B1xZ0iE \mid x \in \{0, 1\}^{+}, i \geq 1\}.$

It is clear that $L(G)$ is contained in the right side.
Consider the reverse inclusion. Since $G$ performs binary addition, it is clear that all words in the first set of the union are in $L(G)$. Consider next an arbitrary word

$$B1xC0^iE, \quad x \in \{0, 1\}^*, \quad i \geq 1. \quad (8.2)$$

By what we already have shown

$$B1x1^iAE \in L(G).$$

On the other hand, a sequence of applications of the production $1A \rightarrow C0$ yields

$$B1x1^iAE \Rightarrow B1xC0^iE,$$

which shows that (8.2) is in $L(G)$.

Consider, finally, an arbitrary word

$$B1xZO^iE, \quad x \in \{0, 1\}^+, \quad i \geq 1. \quad (8.3)$$

Assume first that $x$ does not contain any occurrences of the letter 1, i.e., $x = 0^j, j \geq 1$. Then we obtain

$$B11^{i+j-1}AE \Rightarrow B1C0^{i+j-1}E \Rightarrow B10Z0^{i+j-1}E \Rightarrow B10^jZ0^iE,$$

which shows that (8.3) is in $L(G)$.

Assume, secondly, that $x$ contains an occurrence of 1, i.e., $x = x'10^j$, where $|x'| \geq 0, j \geq 0$. We obtain now

$$B1x'01^{i+j-1}AE \Rightarrow B1x'OC0^{i+j}E \Rightarrow B1x'1Z0^{i+j}E \Rightarrow B1x'10^jZ0^iE,$$

which shows that (8.3) also now is in $L(G)$. Hence we have established (8.1).

Consider now the PCF grammar $H$ with the axioms

$$B10AE, \quad B11AE, \quad B1C0E, \quad B10Z0E, \quad B11Z0E$$

and productions

$$A \rightarrow 0A \mid 1A, \quad C \rightarrow 0C \mid 1C \mid C0, \quad Z \rightarrow 0Z \mid 1Z \mid Z0.$$

Clearly, $H$ generates the right side of (8.1) and, hence, $L(G) = L(H)$. By Theorem 8.2, $L(H)$ is not PCF-monogenic.
9. Generalizations

This final section is to be viewed only as preliminary in nature: the main purpose is to give some definitional suggestions.

The ideas of pure grammars can be extended to concern more general rewriting mechanisms, in particular, Post canonical systems. Consider a basic alphabet $\Sigma$ and an alphabet

$$X = \{x_1, ..., x_n\}$$

of operational variables. A rewriting rule (production) in a Post canonical system $G$ has the form

$$\alpha_1, ..., \alpha_k \rightarrow \beta \quad k \geq 1,$$

(9.1)

where the $\alpha$'s and $\beta$ are words over $\Sigma \cup X$.

The rule (9.1) is applied as follows. For an $n$-tuple of words

$$\gamma = (\gamma_1, ..., \gamma_n)$$

(9.2)

over $\Sigma$, we denote by $\alpha_i(\gamma)$ the result of replacing each occurrence of $x_j$ in $\alpha_i$ with $\gamma_j$, for $1 \leq j \leq n$. $\beta(\gamma)$ is defined similarly. The rule (9.1) can be used to derive, for any $n$-tuple (9.2), the word $\beta(\gamma)$ from the words $\alpha_i(\gamma), ..., \alpha_k(\gamma)$. A Post canonical system $G$ has a finite number of axioms. The language generated by $G$ consists of all words derivable from the axioms by the rewriting rules.

The alphabet $\Sigma$ can be divided into nonterminals and terminals:

$$\Sigma = \Sigma_T \cup \Sigma_N.$$  

(9.3)

In this case, only words over $\Sigma_T$ belong to the generated language. It is well-known that, if the decomposition (9.3) is allowed, then every recursively enumerable language is generated by a Post canonical system, where each of the rules (9.1) is in the normal form

$$\alpha x \rightarrow x \beta, \quad \alpha, \beta \in \Sigma^*.$$

On the other hand, if the decomposition (9.3) is not allowed, i.e., all generated words belong to $L(G)$, then we speak of pure canonical systems. Pure canonical systems have been investigated in Büchi (1974) and Salomaa (1968, 1969).

Observe now that pure grammars can be viewed as pure canonical systems, where every rule has the form

$$x_1 \alpha x_2 \rightarrow x_1 \beta x_2, \quad \alpha, \beta \in \Sigma^*.$$  

(9.4)

Similarly, PCF languages can be viewed as pure canonical systems where every
rule has the form (9.4) where, moreover, \( \alpha \) is a letter of \( \Sigma \). Because of this close connection, the following two theorems due to Kratko, Kratko (1966), are of interest for the theory of pure grammars.

**Theorem 9.1.** Every language generated by a Post canonical system with rules of one of the following two forms

\[
\alpha_1 x_1, \ldots, \alpha_k x_k \rightarrow \beta x, \quad k \geq 1,
\]

\[
xx \rightarrow x^2,
\]

where \( \alpha \)'s and \( \beta \) are words over \( \Sigma, \) is regular. Moreover, every regular language is generated by a pure canonical system with rules of the form (9.6).

**Theorem 9.2.** Every recursively enumerable language is generated by a Post canonical system with rules of one of the following two forms

\[
x_1 x, \alpha_2 x \rightarrow \beta x,
\]

\[
xx_1, xx_2 \rightarrow x^2,
\]

where the \( \alpha \)'s and \( \beta \) are words over \( \Sigma. \) Consequently, pure canonical systems with rules of the forms (9.7) and (9.8) generate nonrecursive languages.

The comparison between rules (9.5)-(9.8) shows that there is a tremendous gap in the generative capacity of Post canonical systems: directly from regular languages to recursively enumerable languages! One possibility to bridge this gap is to impose special conditions on the words \( \alpha \) and \( \beta, \) for instance, to assume that they are letters. This is done in the following definition.

**Definition.** A pure canonical system is **context-free** if all rules are of the form

\[
x_1 a_1 x_2 \cdots x_i a_i x_{i+1} \cdots x_n a_n x_{n+1} \rightarrow x_1 a_1 x_2 \cdots x_i \beta x_{i+1} \cdots x_n a_n x_{n+1},
\]

where each \( a_i \) is a letter of \( \Sigma \) and \( \beta \) is a word over \( \Sigma. \)

Thus, context-free pure canonical systems work like PCF grammars, except that each rule can be applied only in the presence of some specified letters of \( \Sigma, \) and even the order of the letters is specified. Context-free pure canonical systems generate the sets of sentential forms of \( E \)-grammars in the sense of Lombovskaja (1972). (An \( E \)-grammar is a context-free grammar, where each rule can be applied only if some specified nonterminals, depending on the rule, are present.) Consequently, they generate languages which are not context-free. Because of the ordering, they also resemble scattered-context grammars.
ACKNOWLEDGMENT

We express our gratitude to Josef Kaltenbrunner for the inspirational effect of Peterbaueralm on the research reported.

RECEIVED: August 20, 1979; REVISED: September 17, 1979

REFERENCES


Ehrenfeucht, A. H., and Rozenberg, G. (1978), The sequence equivalence problem for OS systems is decidable, manuscript.


Hagauer, J. (1978), A simple variable free CF grammar generating a non regular language, Bull. EATCS 6, 28–33.


