1986

Epsilon Precedence Grammars and Languages

Masoud T. Milani
University of Central Florida

EPSILON PRECEDENCE GRAMMARS AND LANGUAGES

by

MASOUD T. MILANI

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
the Department of Computer Science at
the University of Central Florida
Orlando, Florida

MAY 1986

Major Professor: Dr. David A. Workman

ABSTRACT

The classes of simple and weak precedence grammars are generalized to include ε-rules (productions with the empty right parts). The descriptive power of epsilon simple precedence (ESP) grammars increases directly with the number of ε-rules permitted; the class of ESP grammars with no ε-rules, ESP_0, is identical to the class of simple precedence grammars; ESP grammars with at most one ε-rule, ESP_1, define a class of languages which properly includes the class of ESP_0 languages, but is itself properly included in the class of deterministic context-free languages. In general, ESP grammars having at most i ε-rules, ESP_i, define a class of languages which is properly included in that defined by ESP_{i+1} grammars. This hierarchy of languages exhausts the deterministic context-free languages. The hierarchy of ESP languages is established using an iteration theorem which may be used to show that a given language is not ESP_i for a given i.

An algorithm to convert arbitrary LR(1) grammars to equivalent epsilon weak precedence (EWP) grammars is developed.

The class of Viable Prefix EWP grammars is defined and it is shown that the EWP parser for every Viable Prefix EWP grammar detects syntactic errors at the earliest possible time. Also, it is established that every deterministic context-free language is defined by some Viable Prefix EWP grammar.

Finally, it is shown that the class of EWP grammars, while properly containing the class of Viable Prefix EWP grammars, is itself properly included in the well-known classes of context-free grammars with ε-rules which define exactly the deterministic context-free languages.

ACKNOWLEDGMENTS

I would like to express my deep appreciation for the support and the guidance of my advisor, Dr. David Workman. His contribution to the whole work is fundamental.

I am also grateful to the members of my committee, Drs. Charles Hughes, Ronald Dutton, and Christian Bauer, for their readings and helpful suggestions. I am specially thankful to Dr. Charles Hughes for being my teacher and friend during my years at UCF.

Finally, I would like to thank my parents and my wife, Farah, for their constant love and encouragement that helped me in writing this thesis

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

A language, L, is a set (possibly infinite) of finite length sentences constructed from a finite alphabet Σ. The theory of languages, developed mainly in the past two decades, provides a basis for studying programming languages and the mechanisms appropriate for describing and implementing them. Generally, there are two mechanisms for describing a language: recognizers and generators. A recognizer is a procedure (program or machine) which, when presented an arbitrary sentence over the alphabet, decides whether or not that sentence is a member of the language it defines. A generator is a formalism that describes how to enumerate all and only those sentences that are members of the language being defined. Grammars fall into this latter category. In addition to defining a language precisely, a grammar associates structure with the sentences it generates. It is through this syntactic structure that the semantics of a language can be realized.

Formally, a grammar is a 4-tuple, G=(N,Σ,P,S), where

(1) N is a finite set of symbols called *nonterminals* or *variables*.

(2) Σ is a finite set of *terminal symbols* (symbols of the alphabet).

(3) P is a finite set of ordered pairs (α,β), where α is some string of symbols over the *vocabulary*, $N \cup \Sigma$, containing at least one symbol from N and β is an arbitrary string (perhaps null) over the vocabulary. Elements of P are called *productions* and are usually denoted $\alpha \rightarrow \beta$ for

some α and β. α is called the *left-hand side* or *left part* and β is called the *right-hand side* or *right part* of the production.

(4) S, the start symbol, is a distinguished symbol in N.

(5) $N \cap \Sigma = \phi$.

For example, the grammar G=(N,Σ,P,S), where

S = E
N = { E, T, F}
Σ = { a, b, (,) }
P = { 1: E→ E + T
2: E→ T
3: T→ T * F
4: T→ F
5: F→ (E)
6: F → a
7: F→ b }

defines the set of arithmetic expressions over a, b, *, +, (, and).

The sentences of the language defined by a grammar are generated by starting from the start symbol of the grammar, and repeatedly applying productions until a string consisting only of terminal symbols is generated. Figure 1 shows how (a+b)*a may be generated by G.

SENTENTIAL FORM		RESULT OF THE RULE	RULE APPLIED
E	==>	T	2
T	==>	T * F	3
T * F	==>	T * a	6
T * a	==>	F * a	4
F * a	==>	(E) * a	5
(E) * a	==>	(E + T) * a	1
(E + T) * a	==>	(T + T) * a	2
(T + T) * a	==>	(F + T) * a	4
(F + T) * a	==>	(a + T) * a	6
(a + T) * a	==>	(a + F) * a	4
(a + F) * a	==>	(a + b) * a	7

Figure 1. Generation of (a+b)*a by G.

Chomsky [12], in an attempt to describe natural languages, classified grammars into four categories. Each category was defined by imposing various types of restrictions on the forms of the productions. This classification, known as the Chomsky hierarchy in the literature, is defined below:

Type 0 or *unrestricted grammars*:

No restriction is imposed on the productions.

Type 1 or *context-sensitive grammars*:

The number of symbols in the left-hand side of each production is less than or equal to the number of symbols in its right-hand side.

Type 2 or *context-free grammars*:

The left part of each production consists of a single nonterminal.

Type 3 or *regular grammars*:

Each production is of the form $A \rightarrow tB$ or $A \rightarrow t$, where A and B are in N, and t is a string over Σ.

The different classes of grammars defined in Chomsky's hierarchy have different descriptive powers and are capable of generating different families of languages. The class of languages defined by type $i+1$ grammars is properly included in the class defined by type i grammars. This hierarchy of languages is shown in Figure 2.

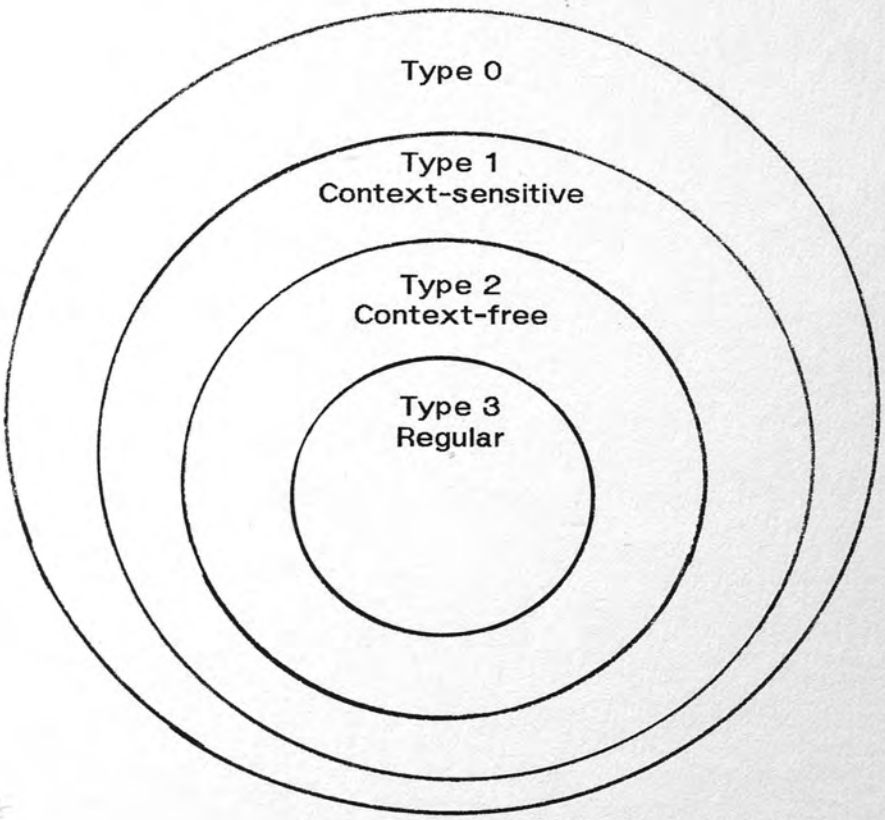
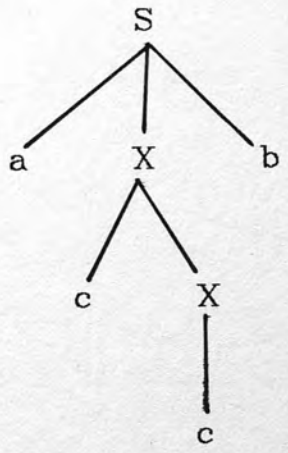


Figure 2. Chomsky Hierarchy.

The first serious attempt to formally define a programming language was made by Backus [8]. He designed a special notation, known as BNF (Backus-Naur Form), to describe the syntactic structure of the ALGOL-60 language - one of the first formally defined programming languages [36]. The formal definition of ALGOL-60 in BNF, which is equivalent in descriptive power to Chomsky's context-free grammars, is considered to be one of the major contributions to computer science. Since the definition of the programming language ALGOL-60, it has been recognized that context-free grammars are well-suited for the formal definition of programming languages.

The definition of a language generally consists of two parts: syntax and semantics. The rules of syntax specify sentences of the language and associate with each sentence a syntactic structure. The rules of semantics associate meaning with each valid syntactic structure. A context-free grammar defines only the syntax of a language. The syntactic structure of a sentence, as defined by some grammar, is usually represented by a labeled tree called the *syntax tree*. For example, consider the grammar G=({S,X},{a,b,c},{S→aXb, X→cX, X→c}, S) describing the language $L=\{ac^n b | n \geq 1\}$. The syntax tree for the sentence "accb" of L is shown below.



A parser is an algorithm designed to reconstruct the syntactic structure of some string presented as input. This can be done in either "top-down" or "bottom-up" fashion. A top-down parser builds the syntax tree starting from the root (start symbol), while a bottom-up parser starts from the leaves. The steps involved in constructing the above syntax tree, both top-down and bottom-up, are shown in figures 3a and 3b, respectively.

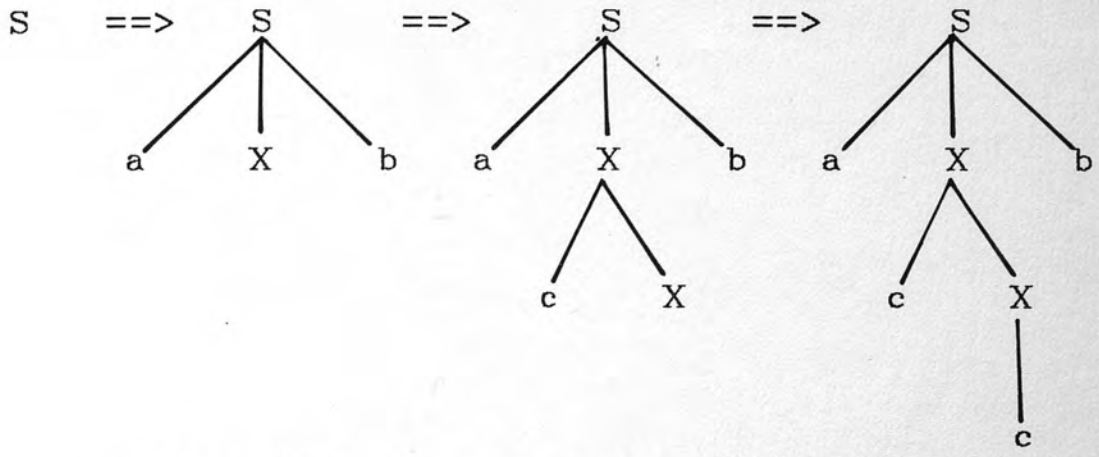


Figure 3a. Top-down generation of a syntax tree.

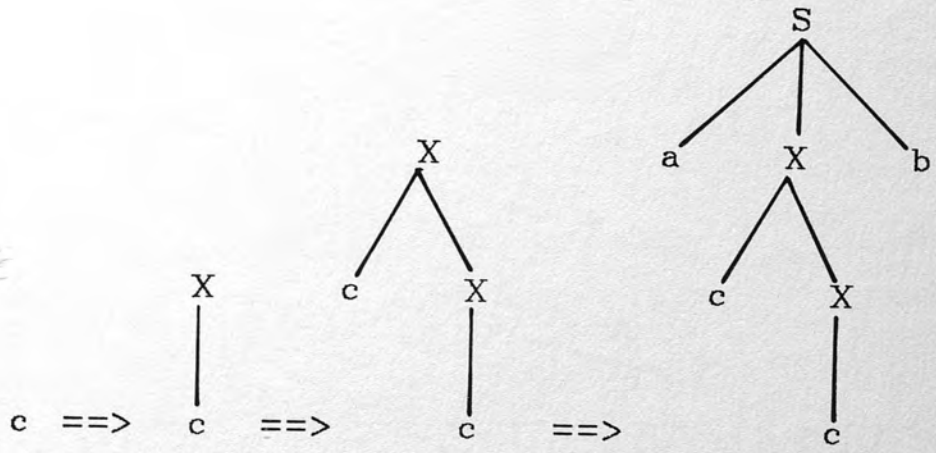


Figure 3b. Bottom-up generation of a syntax tree.

Unfortunately, in spite of a considerable effort, general algorithms to construct efficient parsers (parsing time proportional to the length of input) for arbitrary context-free grammars have not been found [5,22]. Therefore, formal language research has been focused on the development of different subclasses of context-free grammars which may be used to construct efficient parsers. Among the classes developed for this purpose are LL(k), Bounded Right Context

and various types of LR and precedence grammars. Generally, LL(k) grammars are used to build top-down parsers, while Bounded Right Context, LR and precedence grammars yield bottom-up parsers.

At the present time, LR grammars and parsing techniques are the most used device to define programming languages and implement their parsers [4]. This is due to the fact that LR grammars are general and include most of the "natural" grammars for programming languages. Furthermore, LR parsers have good error-detecting capabilities. LR parsers are, however, difficult to construct and their parsing tables consume considerable memory resources.

Precedence parsing techniques, in contrast to LR, enjoy a simple theoretical basis and produce relatively compact parsers. The use of precedence grammars for defining and implementing programming languages, however, is not particularly popular for several reasons.

- Precedence grammars have a small intersection with the class of grammars one would "naturally" write for a programming language.

- Simple and weak precedence grammars, as illustrated in Figure 4, describe a proper subclass of deterministic context-free languages.[1]

- Precedence grammars do not allow ε-rules (productions with the empty right part) which are very convenient for introducing semantic actions at appropriate times during a parse.

---

[1] A context-free language is deterministic if it is accepted by some deterministic pushdown automaton.

- Precedence parsers in general, do not detect syntactical errors at the earliest possible time during a parse.

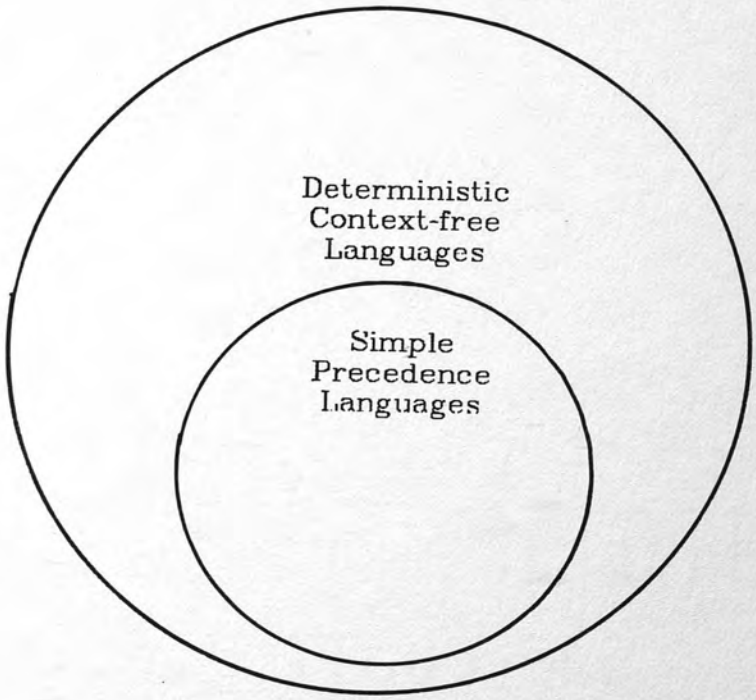


Figure 4. The relationship between deterministic context-free and simple precedence languages.

In the broadest terms, the purpose of the research presented in this dissertation is to further develop precedence techniques. We have extended the class of simple precedence grammars to include ε-rules. This generalization is not trivial since we show that the class of epsilon simple precedence grammars, or ESP, as we call it, describes all deterministic context-free languages. Furthermore, we show that the descriptive power of such grammars increases directly with the number of ε-rules permitted; the class of ESP grammars with no ε-rules, ESP_0, is identical to the class of simple precedence grammars; the ESP grammars with no more than one ε-rule, ESP_1, define a class of languages which properly includes

the class of ESP_0 languages, but is itself properly included in the class of deterministic context-free languages. In general, ESP grammars with at most i ε-rules, denoted ESP_i, define a class of languages which is properly included in that defined by ESP_{i+1} grammars. The hierarchy of languages so defined exhausts the class of deterministic context-free languages. This is the first exhaustive hierarchy of deterministic context-free languages studied in the literature. The other known hierarchies, namely the hierarchies of LL(k) and strict deterministic languages, cover only a proper subset of deterministic context-free languages [5,22,23].

The hierarchy of ESP languages is established using a new iteration theorem; that is, a theorem which characterizes infinite languages of the same class. The iteration theorem is used to show that a given language is not ESP_i, for a fixed i.

We have defined the class of Viable Prefix EWP grammars and shown that the EWP parser constructed for these grammars detects syntactic errors at the earliest possible time. Additionally, an algorithm to convert arbitrary LR(1) grammars to equivalent EWP grammars is developed. We show that the grammars obtained from this algorithm are Viable Prefix EWP grammars. This result shows that it is possible to obtain simple parsers with good error detection capabilities only by restricting the forms of the productions. Such restrictions do not, however, limit the class of languages. As our last result, we show that the well-known classes of context-free grammars with ε-rules defining exactly the class of deterministic context-free languages properly contain the class of EWP grammars.

This dissertation is organized as follows. Chapter 2 contains the relevant terminology, and a review of the literature. In Chapter 3, precedence grammars

are generalized to obtain the classes of epsilon simple, epsilon weak, and extended epsilon weak precedence grammars. A parsing algorithm for epsilon simple precedence grammars is presented in Chapter 3. An algorithm to convert arbitrary LR(1) grammars to equivalent epsilon weak precedence grammars is presented in Chapter 4. Moreover, it is shown in Chapter 4 that the algorithm to convert arbitrary LR(1) grammars to equivalent EWP grammars produces grammars whose epsilon precedence parsers detect syntactical errors at the earliest possible time. Chapter 5 presents an iteration theorem for epsilon simple precedence languages of index i and discusses some of the closure properties of these languages. The hierarchy of the ESP languages is also established in Chapter 5. In Chapter 6, the classes of epsilon simple and weak precedence grammars are compared with the well-known classes of context-free grammars. Finally, Chapter 7 contains our concluding remarks and possible extensions of this research.

CHAPTER 2

BASIC DEFINITIONS AND THE SURVEY OF LITERATURE

In this chapter the topics in parsing theory that are important to our research are surveyed. First, the basic definitions and notations used throughout this thesis are established.

2.1 Basic Terminology

Definitions, examples, lemmas and theorems are numbered sequentially in the order they appear in each chapter. The number designation has the form $s.k$, where s and k denote the chapter number and the occurrence index, respectively. Figures and tables are numbered sequentially starting from one.

For any set of symbols, V, V^* will denote the set of all strings of finite length over V, including the empty string, ε. V^+ denotes $V^*-\{\varepsilon\}$. If α is a string in V^*, α^n denotes the n-fold concatenation of α with itself; $\alpha^0=\varepsilon$, and $\alpha^n=\alpha\alpha^{n-1}$, $n\geq 1$. The length of a string, α, is denoted $|\alpha|$. For all $k\geq 0$, unary operators, $PREF_k$ and $SUFF_k$, are defined on V^* as follows:

$$PREF_k(\alpha)=\begin{cases}\alpha & \text{if } |\alpha|<k\\ \beta & \text{if } \alpha=\beta\delta \text{ and } |\beta|=k\end{cases}$$

$$SUFF_k(\alpha)=\begin{cases}\alpha & \text{if } |\alpha|\leq k\\ \delta & \text{if } \alpha=\beta\delta \text{ and } |\delta|=k\end{cases}$$

A context-free grammar (CFG) is a 4-tuple G=(N,Σ,P,S), where N, Σ, P, and S denote the *nonterminal set*, *terminal alphabet*, *production set* and *start symbol*, respectively. The vocabulary of G, N$\cup\Sigma$, will be denoted by V_G. A production (A,α) is written as A$\rightarrow\alpha$.

If A$\rightarrow\alpha\in P$ and β, $\delta\in V_G^*$, then we say $\beta A\delta$ *directly derives* $\beta\alpha\delta$, or $\beta\alpha\delta$ *directly reduces* to $\beta A\delta$. A direct derivation of $\beta\alpha\delta$ from $\beta A\delta$ is denoted $\beta A\delta$ ==> $\beta\alpha\delta$. A direct derivation, $\beta A\delta$ ==> $\beta\alpha\delta$, is said to be *rightmost* (*leftmost*), denoted $\beta A\delta ==>_{rm}\beta\alpha\delta$ ($\beta A\delta ==>_{lm}\beta\alpha\delta$), if $\delta\in\Sigma^*$ ($\beta\in\Sigma^*$). If $\alpha_1 ==>\alpha_2$, $\alpha_2 ==>\alpha_3$, $\cdots$ $\alpha_i ==>\alpha_{i+1}$, $\cdots$ $\alpha_{k-1} ==>\alpha_k$, $1\leq i<k$, $k>1$, $\alpha_i\in V_G^+$, $\alpha_k\in V_G^*$, then α_k is derived from α_1 and is written as $\alpha_1 ==>^*\alpha_k$. If at least one production is applied in deriving α_k from α_1 then we write $\alpha_1 ==>^+\alpha_k$. A derivation is rightmost (leftmost), if it consists only of direct rightmost (leftmost) derivations. Rightmost (leftmost) analogues of $\alpha_1 ==>^*\alpha_k$ and $\alpha_1 ==>^+\alpha_k$ are denoted $\alpha_1 ==>^*_{rm}\alpha_k$ ($\alpha_1 ==>^*_{lm}\alpha_k$) and $\alpha_1 ==>^+_{rm}\alpha_k$ ($\alpha_1 ==>^+_{lm}\alpha_k$), respectively. A derivation having exactly n steps is denoted $==>^n$. $==>^{*n}$ denotes a derivation having at most n steps.

A string $\alpha\beta\gamma$ is a *sentential form* if $S ==>^*\alpha\beta\gamma$. A sentential form is called a right (left) sentential form if it is derived from S by a rightmost (leftmost) derivation. The string β is said to be the *handle* of a right sentential form $\alpha\beta\gamma$, if there exists a rightmost derivation $S ==>^*\alpha A\gamma ==>\alpha\beta\gamma$. A string $\gamma\in V_G^*$ is called a *viable prefix* of G if $S ==>^*_{rm}\alpha A\omega ==>\alpha\beta\omega$ and $\gamma=PREF_k(\alpha\beta)$, for some $k\geq 0$.

A nonterminal, Z, is said to be *erasable*, if Z$\rightarrow\varepsilon\in$P. The set of erasable nonterminals is denoted N_ε(G). A nonterminal, Z, is said to be *nullable*, if $Z ==>^+\varepsilon$. The set of nullable nonterminals is denoted null(G).

A grammar is said to be *uniquely invertible* (UI), if for all A, B $\in$ N, $A \rightarrow \beta \in$ P and $B \rightarrow \beta \in$ P imply A=B.

For each $\alpha \in V_G^*$, the sets $FIRST_k$, $FOLLOW_k$, and EFF_k are defined as follows:

$$FIRST_k(\alpha)=\{PREF_k(x) \mid \alpha ==>^* x,\ x \in \Sigma^*\}$$

$$FOLLOW_k(\alpha)=\{\omega \mid S ==>^* \alpha\gamma \text{ and } \omega \subset FIRST_k(\gamma)\}$$

$$EFF_k(\alpha)=\begin{cases} PREF_k(\alpha) & \text{if } \alpha \in \Sigma^* \\ PREF_k(x) & \text{if } \textit{there exists a derivation } \alpha ==>_{rm}^* \gamma y ==>_{rm} x \in \Sigma^* \\ & \quad \text{and } \textit{either } \gamma \in \Sigma, \textit{ or } \gamma \in N \text{ and } y \neq x \end{cases}$$

A grammar, G, has a *cycle* if there exists $A \in N$ such that A ==>$^+$A. A grammar, G, is *reduced* if for each production, A $\rightarrow \alpha \subset$P, there exist strings $x, y, z \in \Sigma^*$ such that S==>* xAz ==>$x\,\alpha z$ ==>* xyz. A grammar is said to be *proper* if it is cycle-free and reduced. Let G=(N,Σ,P,S) be a CFG. The *augmented grammar* derived from G is defined to be G'=(N$\cup\{S'\}$,Σ,P$\cup\{S' \rightarrow S\}$, S'), where S' is a new symbol not in V_G.

2.2 Hierarchy of Context-Free Grammars

The more important classes of deterministic context-free grammars and languages are surveyed in this section. The first concept we review is that of LL grammars developed by Rosenkrantz and Stearns [40]. These form the basis for top-down parsers. We then review the concepts related to bottom-up parsing.

DEFINITION 2.1 A CFG, G=(N,Σ,P,S), is said to be LL(k), $k \geq 0$ if the conditions

(1) $S ==>^{*}_{lm} \omega A \beta ==> \omega\alpha\beta ==>^{*} \omega x$

(2) $S ==>^{*}_{lm} \omega A \beta ==> \omega\gamma\beta ==>^{*} \omega y$

(3) $FIRST_k(x) = FIRST_k(y)$

imply $\alpha = \gamma$.

For $k=1$, the grammars satisfying the requirements of the above definition form the basis for deterministic top-down parsers (*recursive descent*). For $k>1$, an LL(k) grammar can be used to construct a recursive descent parser only if it is LL(k) in the strong sense.

DEFINITION 2.2 A CFG, G=(N,Σ,P,S), is said to be LL(k) in the *strong sense*, $k \geq 0$, if for all A∈N such that A→α, A→$\beta \in P$, $\alpha \neq \beta$,

$$FIRST_k(\alpha FOLLOW_k(A)) \cap FIRST_k(\beta FOLLOW_k(A)) = \phi.$$

A recursive descent parser starts from the start symbol, S, and tries to reconstruct a leftmost derivation of the input string by looking at the next k input symbols in each step. In a recursive descent parser, a procedure for each nonterminal is implemented. Each procedure examines the next k input symbols to decide whether to advance the input, call another procedure, or terminate. Conditions of LL grammars guarantee that the next action in each procedure may be uniquely determined. LL(k) languages form a proper subset of the deterministic context-free languages. This is shown by proving that the language $L=\{0^n 1^n \mid n \geq 1\} \cup \{0^n 2^n \mid n \geq 1\}$ is not described by any LL grammar [5]. Furthermore Kurki-Suonio [29] established that for all $k>0$, the class of LL(k-1)

languages is properly included within the class of LL(k) languages. This hierarchy of LL(k) languages is established by showing that for all $k>0$, the language

$$L_k=\{a^n\omega \mid n\geq 1,\ \omega\in\{b,c,b^k d\}^n\}$$

is an LL(k) language which is not LL(k-1).

In general, a bottom-up (shift-reduce) parser consists of a stack and an input buffer. The parser "shifts" the input symbols onto the stack until a handle appears on top of the stack. The handle is then replaced by the left part of the appropriate production. This process is repeated until either an error configuration is reached or the input buffer becomes empty and the stack contains only the start symbol of the underlying grammar.

There are many grammar classes studied in the literature which restrict the forms of the productions to facilitate the implementation of bottom-up parsers. We start our survey of bottom-up grammars by the class of Bounded Right Context (BRC) grammars introduced by Floyd [16].

DEFINITION 2.3 A CFG, G=(N,Σ,P,S), is said to be (m,n)*Bounded Right Context* (BRC) for $m,n\geq 1$ if in the augmented grammar, G', the conditions

(1) $\$^m S'\$^n \Rightarrow_{rm}^{*} \vartheta A\omega \Rightarrow_{rm} \vartheta\beta\omega$,

(2) $\$^m S'\$^n \Rightarrow_{rm}^{*} \gamma Bx \Rightarrow_{rm} \gamma\delta x = \vartheta'\beta y$,

(3) $|x|\leq|y|$,

(4) $SUFF_m(\vartheta')=SUFF_m(\vartheta)$ and $PREF_n(y)=PREF_n(\omega)$

imply $\vartheta' Ay=\gamma Bx$; that is, $\vartheta'=\gamma$, $y=x$ and A=B.

The conditions of (m,n)BRC grammars guarantee that in each step during a parse, the next action of the parser may be uniquely determined by examining the

next n input symbols and at most $m+l$ symbols of the stack where l is the length of the longest right part in the grammar. BRC parsers are quite large and difficult to construct. (1,1)BRC grammars define exactly the class of deterministic context-free languages [5].

Fundamental to our research are simple precedence grammars and languages introduced by Wirth and Weber [43]. These grammars are defined in terms of precedence relations defined below.

DEFINITION 2.4 Let G=(N,Σ,P,S) be a CFG without ε-rules. For each X∈N define sets LEFT(X), RIGHT(X) as follows:

LEFT(X)=$\{Z \in V_G \mid X \Rightarrow^{+}_{lm} Z\vartheta$, for some $\vartheta \in V_G^*\}$,

RIGHT(X)=$\{Z \in V_G \mid X \Rightarrow^{+}_{rm} \vartheta Z$, for some $\vartheta \in V_G^*\}$.

Then, for X,Y$\in V_G$ and $t \in \Sigma$ define the *precedence relations* $<\cdot$, $\doteq$, $\cdot>$ on V_G as follows:

(1) X$\doteq$Y, if there exists a production A$\rightarrow\alpha$XY$\beta\in$P for some $\alpha\beta \in V_G^*$;

(2) X$<\cdot$Y, if there exists Z⊂N such that X$\doteq$Z and Y⊂LEFT(Z);

(3) X$\cdot>t$, if there exists Z∈N such that Z$\doteq t$ and X∈RIGHT(Z) or, if there are Z_1, Z_2∈N where $Z_1 \doteq Z_2$ with X∈RIGHT(Z_1) and $t \in$LEFT(Z_2).

The purpose of the precedence relations was 1) to identify what pairs of symbols could legally appear in a viable prefix and 2) to classify these pairs according to whether they defined the left end of the handle ($<\cdot$), the right end of the handle ($\cdot>$) or occurred within the handle ($\doteq$) of a right sentential form. These relations were then used to define a class of grammars that could be deterministically parsed bottom-up.

DEFINITION 2.5 Let G=(N,Σ,P,S) be a reduced CFG without ε-rules. G is said to be *simple precedence* if the following conditions hold.

(1) G has no cycles;

(2) G is uniquely invertible;

(3) The precedence relations, $<\cdot$, $\doteq$, $\cdot>$ are pairwise disjoint.

In parsing algorithms based on simple precedence grammars the precedence relations were extended to include a special symbol, \$, which is commonly used to denote the end of the input string to be parsed as well as the bottom of the parse stack. The precedence relations are stored in a matrix called the precedence matrix. Each row and column of this matrix represents a symbol of the vocabulary of the grammar. The entry in row i, and column j of the precedence matrix contains the precedence relation which holds between the symbols represented by i and j, respectively. The algorithm functions by shifting input symbols onto the stack as long as the relations $<\cdot$ or $\doteq$ hold between the stack top and the next input symbol. A stack reduction is initiated when the relation, $\cdot>$, holds. If neither of these relations hold, then an error is reported. When stack reductions are made the relation, $<\cdot$ is used to locate the left end of the handle in the stack. The stack is reduced by the production whose right part appears on top of the stack.

Wirth and Weber [43] suggest that, for some precedence grammars, the precedence relations can be represented by two vectors, f and g, called precedence functions such that

$$\begin{aligned} f(X) = g(Y) &\quad \text{implies } X \doteq Y \\ f(X) < g(Y) &\quad \text{implies } X <\cdot Y \\ f(X) > g(Y) &\quad \text{implies } X \cdot> Y \end{aligned}$$

The use of precedence functions instead of a precedence matrix reduces the storage requirements of a precedence parser. Precedence functions, however, do not exist for every precedence grammar. The use of precedence functions, in general, decreases the error detecting power of precedence parsers [5].

Since a precedence parser shifts when either $<\cdot$ or $\doteq$ holds between the stack top and the next input, the requirement that $<\cdot$ and $\doteq$ be disjoint can be relaxed. This observation lead to the notion of weak precedence grammars originally due to Ichbiah and Morse [24].

DEFINITION 2.6 Let G=(N,Σ,P,S) be a reduced CFG without ε-rules. G is said to be ***weak precedence*** (WP) if the following conditions hold:

(1) G is cycle-free;

(2) G is uniquely invertible;

(3) The Wirth-Weber precedence relations, $<\cdot$, $\doteq$, $\cdot>$ satisfy $(<\cdot \cup \doteq) \cap \cdot> = \phi$;

(4) if $X \rightarrow \alpha A \beta$ and $Y \rightarrow \beta$ belong to P, then $(A,Y) \notin (<\cdot \cup \doteq)$.

In [2] it is established that weak precedence grammars are equivalent in language power to the class of simple precedence grammars. Fischer [15] has shown the simple precedence languages to be a proper subclass of the deterministic context-free languages by demonstrating that the set $L=\{a\,0^n\,1^n \mid n>0\} \cup \{b\,0^n\,1^{2\times n} \mid n>0\}$ is not a simple precedence language.

Simple precedence grammars can be generalized by increasing the number of symbols used to define the precedence relations. This gave rise to the notion of extended precedence relations and grammars credited to Gray [20].

DEFINITION 2.7 Let G=(N,Σ,P,S) be a reduced CFG without ε-rules. For $m,n \geq 1$ define the (m,n)-*precedence relations*, $<\cdot$, $\doteq$, $\cdot>$, on $(\{\$\} \cup V_G)^+$ as follows. Let $\$^m S\$^n ==>^*_{rm} \vartheta X \omega ==>_{rm} \vartheta\delta\omega$ be any rightmost derivation, then

(1) $SUFF_m(\vartheta) <\cdot v$, $v \subset \{PREF_n(\delta\omega)\} \cup \{FIRST_n(\delta\omega) \mid PREF_1(\delta) \subset \Sigma\}$;

(2) for each $\delta_1 \neq \varepsilon$ and $\delta_2 \neq \varepsilon$ such that $\delta = \delta_1\delta_2$ define $SUFF_m(\vartheta\delta_1) \doteq PREF_n(\delta_2\omega)$, and if δ_2 begins with a terminal define $SUFF_m(\vartheta\delta_1) \doteq v$, $v \in FIRST_n(\delta_2\omega)$;

(3) $SUFF_m(\vartheta\delta) \cdot > PREF_n(\omega)$.

G is said to be (m,n)-*precedence parsable* if

(4) G is cycle-free;

(5) G is uniquely invertible;

(6) the (m,n)-precedence relations $<\cdot$, $\doteq$, $\cdot>$, are pairwise disjoint for G.

A corollary to this definition is that every simple precedence grammar is (1,1)-precedence parsable.

Graham [18] showed that every deterministic context-free language is described by some (2,1)-precedence parsable grammar.

In the same manner that simple precedence grammars were generalized to (m,n)-precedence parsable grammars, weak precedence grammars can be generalized to (m,n)-weak precedence grammars. The extension to weak precedence grammars presented in our next definition was introduced in [44].

DEFINITION 2.8 Let G=(N,Σ,P,S) be a reduced CFG without ε-rules. G is said to be (m,n)-*weak precedence* for $m,n \geq 1$ if

(1) G is cycle-free;

(2) G is uniquely invertible;

(3) the (m,n)-precedence relations $<\cdot$, $\doteq$, $\cdot>$ for G satisfy, $(<\cdot \cup \doteq) \cap \cdot> = \phi$;

(4) If $X \rightarrow \alpha\beta$ and $Y \rightarrow \beta$ are distinct productions, then for no γ, $\delta \in (V_G \cup \{\$\})^*$ is it true that $(SUFF_m(\gamma\alpha), PREF_n(Y\delta)) \in (<\cdot \cup \doteq)$.

In general, writing a precedence grammar for a given language is not an easy task due, in part, to the requirement of unique invertibility. McKeeman [34], in an attempt to relax this requirement, developed the class of Mixed Strategy grammars.

DEFINITION 2.9 Let G=(N,Σ,P,S) be a reduced CFG without ε-rules. G is said to be Simple Mixed Strategy Precedence (SMSP) if

(1) G is cycle-free;

(2) Wirth-Weber relation $\cdot>$ is disjoint from the union of the relations $\doteq$ and $<\cdot$;

(3) let $l(A)=\{X \mid (X,A) \in <\cdot \cup \doteq\}$, then

If $A \rightarrow \beta' X \beta$ and $B \rightarrow \beta$ are two productions in P, then X is not in $l(B)$;

If $A \rightarrow \beta$ and $B \rightarrow \beta$ are two productions in P, $A \neq B$, then $l(A) \cap l(B) = \phi$.

An SMSP parsing algorithm is identical to that of weak precedence, except that once the handle is identified, one symbol below the handle (condition (3)) is used to uniquely determine the production to be used in the reduction.

We now review the class of LR(1) grammars and parsers due to Knuth [26].

DEFINITION 2.10 A reduced CFG, G=(N,Σ,P,S), is said to be LR(k) for $k \geq 0$, if in the augmented grammar, G', the conditions

(1) $S' ==>_{rm}^{*} \alpha A \omega ==>_{rm} \alpha\beta\omega$,

(2) $S' ==>_{rm}^{*} \gamma B x ==>_{rm} \alpha\beta y$

(3) $FIRST_k(\omega) = FIRST_k(y)$,

imply $\alpha=\gamma$, A=B, and $x=y$.

LR grammars are the most general class of grammars defining deterministic context-free languages. However, LR parsers are difficult to construct without the use of automatic parser generating systems and can consume considerable memory resources. The construction of LR parsers is covered extensively in [3,4,5,22]. Here, we explain the behavior of LR parsers. First, the notion of LR(k) items is introduced.

DEFINITION 2.11 Let G=(N,Σ,P,S) be a CFG. We say that $[A \rightarrow \beta_1 . \beta_2 \mid u]$ is an LR(k) item for G, if $A \rightarrow \beta_1\beta_2 \in P$, and $u \in \Sigma^k$. $[A \rightarrow \beta_1 . \beta_2 \mid u]$ is said to be valid for $\alpha\beta_1$, a viable prefix of G, if there exists a derivation $S ==>_{rm}^{*} \alpha A \omega ==>_{rm} \alpha\beta_1\beta_2\omega$ in G such that $u \in FIRST_k(\omega)$.

An LR(k) item, $[A \rightarrow \beta_1 . \beta_2 \mid u]$, indicates that at some stage during a parse, we have seen a string derivable from β_1 and expect to see a string derivable from β_2, and $FIRST_k(\beta_2 u)$ is the acceptable input lookahead.

The *canonical collection* of LR(k) items for a grammar, G, defined below

$$\{\{I \mid I \text{ is a valid LR}(k) \text{ item for } \gamma\} \mid \gamma \text{ is a viable prefix of G}\}$$

forms the basis for implementation of LR(k) parsers. Each set of items in the canonical collection of LR(k) items is represented by one state of a Deterministic Finite Automaton (DFA), known as the GOTO graph. This DFA recognizes viable

prefixes of the underlying grammar. Two functions called ACTION and GOTO tables are used by LR parsers. The GOTO function is essentially the transition function of the GOTO graph. It takes a state and a grammar symbol as input and returns a state. For each state of the LR parser, I, each $u \in \Sigma^k$, ACTION (I,u) may have one of the following values:

- shift;
- reduce;
- accept;
- error.

Each stack entry of the LR parser is a pair (I,X), where I is a state and X is a grammar symbol. Initially, the pair (I_0,ε) is pushed onto the stack where I_0 is the initial state of the parser. Let (I,X) be the top stack and u be the next k input symbols. The behaviour of the LR parser is then summarized as follows:

(1) If ACTION(I,u) = shift, then the entry (GOTO(I,$PREF_1(u)$),$PREF_1(u)$) is shifted onto the stack.

(2) If ACTION(I,u) = reduce A→α, where $|\alpha| = n$, then n entries are popped from the stack and the entry (GOTO(J,A),A) is pushed onto the stack where (J,Y) is the $n+1^{st}$ entry in the stack.

(3) If ACTION(I,u) = accept, then the input is accepted as a valid sentence. (Note that in this case u is $\$^k$.)

(4) If ACTION(I,u) = error, then an error is announced.

The conditions of LR grammars guarantee that every entry of ACTION and GOTO tables is uniquely defined.

EXAMPLE 2.12 The ACTION and GOTO tables of the LR(1) parser for the grammar

S → a X b
X → c X
X → c

are shown in Table 1.

STATE	ACTION				GOTO				
	a	b	c	$	a	b	c	X	S
I_0	shift				I_1				I_6
I_1			shift				I_2	I_4	
I_2		reduce $X \rightarrow c$	shift				I_2	I_3	
I_3		reduce $X \rightarrow cX$							
I_4		shift				I_5			
I_5				reduce $S \rightarrow aXb$					
I_6				accept					

Table 1. ACTION and GOTO tables of an LR parser.

The sequence of moves of the LR parser on the input "accb" is shown in Figure 5.

STACK	INPUT
(I_0,ε)	a c c b $
$(I_0,\varepsilon)\,(I_1,a)$	c c b $
$(I_0,\varepsilon)\,(I_1,a)\,(I_2,c)$	c b $
$(I_0,\varepsilon)\,(I_1,a)\,(I_2,c)\,(I_2,c)$	b $
$(I_0,\varepsilon)\,(I_1,a)\,(I_2,c)\,(I_3,X)$	b $
$(I_0,\varepsilon)\,(I_1,a)\,(I_4,X)$	b $
$(I_0,\varepsilon)\,(I_1,a)\,(I_4,X)\,(I_5,b)$	$
$(I_0,\varepsilon)\,(I_6,S)$	$

Figure 5. Moves of the LR parser on input "accb."

The next concept we review is that of SR(s,k) grammars and parsers, due to Workman [44]. An SR parser is similar to an LR(k) parser except that the states of an SR(s,k) parser incorporate information determined by only s symbols of the stack.

DEFINITION 2.13 An LR(k) item, I=[X→$\alpha.\beta \mid u$], is said to be *s-consistent* with γ, $s \geq 0$, if I is valid for some viable prefix $\vartheta\alpha$, where $\gamma = SUFF_s(\vartheta\alpha)$.

For given values of s, k and $\gamma \in V^{*s}$, $\Lambda_{s,k}^{G}(\gamma)$ denotes the (s,k)-*consistency* set for γ containing all k-items, s-consistent with γ. $\mathbf{C}_{s,k}(G)$ denotes the collection of all non-empty (s,k)-consistency sets, $\Lambda(\gamma)$, for G. $\mathbf{C}_{s,k}(G)$ is called the canonical collection of SR(s,k) items for G. In [44] it is established that each (s,k)-consistency set, $\Lambda_{s,k}(\gamma)$, is the union of LR(k) states, $\Gamma_k(\vartheta)$, where ϑ is a viable prefix satisfying $\gamma = SUFF_s(\vartheta)$.

Having formed the canonical collection of (s,k)-consistency sets one can construct a shift-reduce parser using essentially the same techniques employed in constructing a canonical LR(k) parser. A grammar, G, is said to be SR(s,k) precisely when its canonical parsers constructed with states, $\Lambda_{s,k}(\gamma) \in \mathbf{C}_{s,k}(G)$, is deterministic according to the definition below.

DEFINITION 2.14 G=(N,Σ,P,S) is said to be SR(s,k) if

(1) [X→$\alpha.\beta \mid u$], [Y→$\delta. \mid v$] $\in \Lambda(\gamma) \in \mathbf{C}_{s,k}(G)$ imply $v \notin EFF_k(\beta u)$;

(2) [X→$\alpha\beta. \mid u$], [Y→$\beta. \mid u$] $\in \Lambda(\gamma) \in \mathbf{C}_{s,k}(G)$ imply [X→$\alpha.\beta \mid u$], and [Y→$.\beta \mid u$] do not both belong to the same (s,k)-consistency set.

(3) [X→$\alpha. \mid \varepsilon$] $\in \Lambda(SUFF_s(S))$ implies that X=S' and α= S.

It is shown in [44] that every (m,n)-weak precedence grammar is SR(m,n) and every SR(m,n) grammar is (m,n)BRC.

We conclude this section with a review of the class of strict deterministic languages [23]. This is the class of prefix-free languages accepted by a deterministic pushdown automaton in a final state with the empty stack. First, the notion of deterministic pushdown automata is introduced.

DEFINITION 2.15 A deterministic pushdown automaton (DPDA) is a 7-tuple $P=(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$, where

(1) Q is a finite set of states;

(2) Σ is a finite set of input symbols;

(3) Γ is a finite set of stack symbols;

(4) $q_0 \in Q$ is the initial state;

(5) $Z_0 \in \Gamma$ is the initial stack symbol;

(6) $F \subseteq Q$ is the set of final states

(7) δ is a mapping from $Q\times(\Sigma \cup \{\varepsilon\})\times\Gamma$ to the finite subsets of $Q\times\Gamma^*$ such that for each $q\in Q$, $Z\in\Gamma$ and $a \in \Sigma$ either of the following holds

(7.1) $\delta(q,a,Z)$ contains at most one element and $\delta(q,\varepsilon,Z)=\phi$,

(7.2) $\delta(q,a,Z)=\phi$, and $\delta(q,\varepsilon,Z)$ contains at most one element.

A configuration of P is a triple (q,ω,α) in $Q\times\Sigma^*\times\Gamma^*$, where

(1) q is the current state;

(2) ω is the unscanned portion of the input.

A move by P is represented by the relation |--- on configurations.

We are now ready to formally define the class of strict deterministic languages.

DEFINITION 2.16 A language, L, is said to be strict deterministic language of degree n, if there exists a DPDA, $P=(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$, such that $|Q|=n$, and for all $\omega \in L$

$$(q_0,\omega,Z_0) \overset{*}{\vdash} (q,\varepsilon,\varepsilon), \quad q\in F \quad \text{if and only if} \quad \omega\in L.$$

The class of strict deterministic languages is equivalent to the class of LR(0) languages which is properly included in the class of deterministic context-free languages. It is shown in [23] that for all $n\geq 1$, the class of strict deterministic languages of degree n is properly included in the class of strict deterministic languages of degree $n+1$.

The hierarchy of the classes of context-free grammars studied in this chapter is shown in Figure 6.

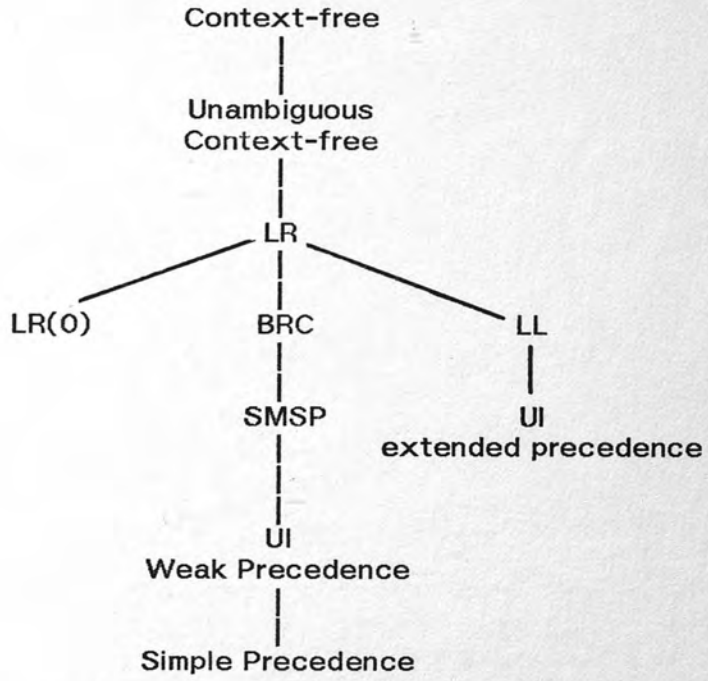


Figure 6. The hierarchy of context-free grammars.

2.3 Precedence Parsers and Correct Prefix Property

Ideally, a parser should detect syntactic errors before the erroneous input symbol is shifted onto the parse stack. Formally, we say that a parsing method has the *Correct Prefix Property* if the existence of a syntax error is detected as soon as the input scanned no longer forms a prefix of any sentence in the language being parsed [19]. Unfortunately, precedence parsing techniques do not have this property. For example, consider the following simple precedence grammar:

$$S \rightarrow a\ x\ y$$
$$S \rightarrow b\ x\ z$$

L(G) consists of sentences axy and bxz. Now consider the moves made by the parser acting upon the erroneous input axz.

$$
\begin{aligned}
(\$,axz\$) &\vdash \xrightarrow{S} (\$a,xz\$) \\
&\vdash \xrightarrow{S} (\$ax,z\$) \\
&\vdash \xrightarrow{R} (\$axz,\$) \\
&\vdash \text{ERROR}
\end{aligned}
$$

The parser did not detect the error in configuration (ax,z). Instead, it shifted "z." This was due to the relation $\doteq$ that holds between x and z through the production S→b x z.

There has been some research [19,30,35] in improving the error-detecting capabilities of precedence parsers. Leinius [30] defined three syntactic error categories in the context of precedence parsing:

1) Character-pair error:

   No precedence relation holds between the top stack symbol and the next input symbol.

2) Phrase error:

   The precedence relation $\cdot>$ holds between the top stack symbol and the next input symbol. But the right part of no production matches the string identified by the reduce function as the handle. The error in the previous example falls into this category.

3) Stackability error:

   The precedence relation $\cdot>$ holds between the top stack symbol and the next input symbol. Although a production A→α, is identified as the production to be used for the reduction, no precedence relation holds between the symbol below α and A.

The simple precedence parser designed by Leinius detects stackability errors by replacing the handle, α, by A, where A→α is a production, only after

determining that A is related to the symbol below α in the parse stack. Consider the following simple precedence grammar:

S → a A e
S → b B e
A → x y
B → x z

Leinius's parser detects stackability error in the input "bxye" after shifting the the symbol "y." A traditional simple precedence parser, however, will announce an error after shifting the last symbol of the input string.

Graham and Rhodes [19] developed a simple precedence parser that has better error detecting capabilities than that of Leinius. In addition to three types of errors defined by Leinius, their parser detects Right Hand Side (RHS) errors. A RHS error occurs when the relation $\doteq$ holds between the top stack symbol and the next input symbol, but after shifting the input symbol, the prefix on the top of the stack is no longer a prefix of right part of any production. Consider the grammar

S → a A
S → x b y
A → b c d

The actions of Graham and Rhodes' parser acting upon the input "aby" is shown below:

($,aby$) $\vdash_{S}$ (a,by)
$\vdash_{S}$ (ab,y)
$\vdash$ RHS ERROR

Although the relation $\doteq$ holds between b and y, the parser did not shift y because "by" is not a prefix of the right part of any production in the grammar. RHS errors are efficiently detected by conducting an incremental search of the right parts.

Moll [35], developed the class of Left Context Precedence grammars, LCP, that apparently result in small and simple parsers with the correct prefix property. Moll, however, did not give an algorithm to decide whether a given algorithm is LCP. Moll's parser announces a syntactic error, if it is determined during a parse, that the underlying grammar is not LCP. Although Moll shows that the class of LCP languages is larger than the class of simple precedence languages, he states the equivalence of LCP and deterministic context-free languages as an open problem. The result of the following theorem is used to define LCP grammars.

THEOREM 2.17 Let G=(N,Σ,P,S) be a precedence grammar, and let $\gamma \in V^+$ be a viable prefix of G. There exists a unique sequence, $\psi_i, \in V^+, i=1, \cdots, n$ such that:

(1) $\gamma=\psi_1 \cdots \psi_n$;

(2) $SUFF_1(\psi_i) <\cdot PREF_1(\psi_{i+1}), i=1, \cdots, n-1$;

(3) There exist $\overline{\psi}_i \in V^*$ and A∈N such that: $A_i \rightarrow \psi_i \overline{\psi}_i \subset$ P, $i=1, \cdots, n$.

DEFINITION 2.18 Let G=(N,Σ,P,S) be a precedence grammar, and let $\gamma \in V^+$ be a viable prefix of G. For $\gamma \in V^+$, the sequence, $\psi_i, i=1, \cdots, n$ of the Theorem 2.17 is called a prefix representation of γ. ψ_n is called the actual prefix of γ. The sequence $\psi_i \in V^+, i=0, \cdots, n$ is called a prefix representation of $\$\gamma$ if

(1) ψ_0=$;

(2) $\psi_i \in V^+, i=1, \cdots, n$ is a prefix representation of γ.

DEFINITION 2.19 Let G=(N,Σ,P,S) be a precedence grammar, and let $\gamma \in V^+$ be a viable prefix of G with prefix representation $\psi_0 \cdots \psi_n$ of $\$\gamma$.

(1) The set GL(ψ_0)=GL(\$)={S} is called the set of goal variables of ψ_0.

(2) For $i=1,\cdots,n$ the set GL($\psi_0 \cdots \psi_i$)= {A$\in N$ | there exists $\$S\$ ==>^*_{rm} \psi_0\psi_1 \cdots \psi_{i-1}Az\$ ==>^*_{rm} \psi_0\psi_1 \cdots \psi_i\lambda_i z\$, z\in\Sigma^*, \lambda_i \in V^*\}$ is called the set of goal variables for $\$\gamma$.

DEFINITION 2.20 A precedence grammar G is said to be Left Context Precedence (LCP) if the precedence parsing function VP defined below, is uniquely defined for G.

$$VP(\$\gamma,z\$)=\begin{cases} (\$\gamma a, z'\$) & \text{if } z=az',\ SUFF_1(\gamma)\doteq a \text{ and } \textit{there exists } A\in GL(\psi_0\cdots\psi_n) \textit{ such that } A\to\psi_n a\lambda_n\in P, \text{for } \textit{some } \lambda_n\in V^* \\ (\$\gamma a, z'\$) & \text{if } z=az',\ SUFF_1(\gamma)<\cdot a \text{ and } \textit{there exists } B\in GL(\psi_0\cdots\psi_n) \textit{ such that } B\to\psi_n C\lambda_n\in P \text{ and } C==>^+_{rm} a\lambda\in P, \text{for } \textit{some } \lambda,\lambda_n \textit{ in } V^* \\ (\psi_0\cdots\psi_{n-1}B, z\$) & \text{if } SUFF_1(\gamma)\cdot>PREF_1(z) \text{ and } \textit{there exists } B\in GL(\psi_0\cdots\psi_n) \textit{ such that } B\to\psi_n\in P \\ \textit{UNDEFINED} & \textit{OTHERWISE} \end{cases}$$

where γ is a viable prefix of G and $\psi_0 \cdots \psi_n$ is a prefix representation of $\$\gamma$.

2.4 Iteration Theorems

Iteration theorems are among the most powerful tools which can be used to show that languages are not defined by any grammar in a particular class.

Iteration theorems usually state necessary conditions that, if satisfied by some sentence of a language, guarantee the membership of a set (usually infinite) of sentences in that language. Iteration theorems exist for context-free [9], deterministic context-free [37] LL(k) [10], strict deterministic [25], and simple precedence languages [28].

Iteration theorems for context-free (pumping lemma) and simple precedence languages are reviewed in this section.

THEOREM 2.21 (Iteration theorem for context-free languages)

Let L=L(G) be a context-free language, where G=(N,Σ,P,S). There exists a constant n, such that if $z \in$L has length greater than n, then there exists a factorization (G-factorization) $z = uv\,\omega xy$ such that

(1) there exist A$\in N$, α, $\beta \in V_G^*$, such that

$S ==>_{rm}^{*} \alpha Ay$; $A ==>_{rm}^{*} \beta Ax$; $A ==>_{rm}^{*} \omega$; $\alpha ==>_{rm}^{*} u$;

$\beta ==>_{rm}^{*} v$;

(2) for every $i \geq 0$, $uv^i \omega x^i y \in$L;

(3) $|v\,\omega x| \leq n$, and $|vx| > 0$.

The essential idea behind Theorem 2.21 is that, if a sentence $z \in$L(G) is long enough, then in any derivation for z, some nonterminal, A, must be repeated at least once. Thus, S$==>_{rm}^{*} \alpha Ay$, and $A ==>_{rm}^{*} \beta Ax$, and therefore, for all $i \geq 0$,

$$S ==>_{rm}^{*} \alpha Ay ==>_{rm}^{*} \alpha\beta Axy ==>_{rm}^{*} \cdots ==>_{rm}^{*} \alpha\beta^i Ax^i y ==>_{rm}^{*} uv^i \omega x^i y.$$

THEOREM 2.22 (Iteration theorem for simple precedence languages)

Let L=L(G) be a simple precedence language, where G=(N,Σ,P,S) is a simple precedence grammar. Suppose that there exists s_1, $s_2 \in L$ with G-factorizations of $s_1 = u_1 v_1 \omega_1 x_1 y_1$ and $s_2 = u_2 v_2 \omega_2 x_2 y_2$, such that $v_1, v_2 \in a^+$, for some $a \in \Sigma^+$, and there exists $r \geq 0$ and $z \in \Sigma^*$, such that

(1) $PREF_1(z) = PREF_1(x_2 y_2)$;

(2) $u_1 v_1^r v_2^2 w_2 x_2 z \in L$.

Then, for all $m \geq 0$, $u_1 v_1^r v_2^{m+1} w_2 x_2^m z \in L$.

CHAPTER 3

EPSILON PRECEDENCE GRAMMARS AND LANGUAGES

The classes of epsilon simple precedence, epsilon weak precedence and extended epsilon weak precedence grammars and languages are developed in this chapter. The class of epsilon simple precedence grammars is obtained by extending the class of simple precedence grammars to include ε-rules. We show that the class of epsilon simple precedence languages, properly includes the class of simple precedence languages.

The class of epsilon simple precedence grammars is further generalized resulting in another new class of grammars which we call epsilon weak precedence grammars. The class of languages defined by epsilon weak precedence languages, however, is shown to be equivalent to that defined by epsilon simple precedence grammars. Finally, the class of epsilon weak precedence grammars are generalized to obtain the class of extended epsilon weak precedence grammars.

Before formally defining these classes of grammars and languages, we present the following concepts.

DEFINITION 3.1. Let G=(N, Σ ,P ,S) be a CFG. For each X $\in$ N, we define the sets L, $\mathbf{L}_\varepsilon$, R, $\mathbf{R}_\varepsilon$ as follows:

$\mathbf{L}(X)$= {Z | there is a leftmost derivation $\pi \in P^+$ such that $X ==>_{lm}^{\pi} Z\alpha$, $Z \in V_G$, $\alpha \in V_G^*$ and for all $(n: Y \to \varepsilon) \in P$, $n \notin \pi$}

$\mathbf{L}_\varepsilon(X)$= {Z | there exists a leftmost derivation $\pi n \pi' \in P^+$ such that $(n: Y \to \varepsilon) \in$P and $X ==>_{lm}^{\pi} Y\Theta ==>_{lm}^{n} \Theta ==>_{lm}^{\pi'} Z\alpha$, where $Z \in V_G$, $\Theta \in V_G^+$, $\alpha \in V_G^*$}

R(X)= {Z | there is a rightmost derivation $\pi \in P^+$ such that $X ==>_{rm}^{\pi} \alpha Z$, $Z \in V_G$, $\alpha \in V_G^*$, and for all $(n: Y \to \varepsilon) \in P$, $n \notin \pi$}

$\mathbf{R}_\varepsilon$(X)= {Z | there exists a rightmost derivation $\pi n \pi' \in P^+$ such that $(n: Y \to \varepsilon) \in$P and $X ==>_{rm}^{\pi} \Theta Y ==>_{rm}^{n} \Theta ==>_{rm}^{\pi'} \alpha Z$, where $Z \in V_G$, $\Theta \in V_G^+$, $\alpha \in V_G^*$}

The set **R** (**L**) of a nonterminal, X, contains the symbols that may appear rightmost (leftmost) in a rightmost (leftmost) derivation $X ==>^+ \alpha$, which does not apply epsilon rules. On the other hand, the set $\mathbf{R}_\varepsilon$ ($\mathbf{L}_\varepsilon$) for X contains the symbols that may appear rightmost (leftmost) in a rightmost (leftmost) derivation $X ==>^+ \alpha$, that does apply epsilon rules.

Just as Wirth-Weber relations, we define epsilon precedence relations. These relations are later used to define different classes of epsilon precedence grammars.

DEFINITION 3.2 The *Epsilon Precedence* (EP) *relations* $\lessdot$, $\doteq$, $\gtrdot$ for a CFG, G=(N,Σ,P,S), are defined as follows, where X,Y∈V_G and $t \in \Sigma$:

(1) $X \doteq Y$ if $A \to \alpha XY\beta \subset P$.

(2) $X \lessdot Y$ if there exists $Z \in N$ such that Y ∈**L**(Z) and $X \doteq Z$.

(3) Define $X \overset{\varepsilon}{=} Y$ if for $n \geq 1$, there exist $Z_1, Z_2, \cdots, Z_n$, in null(G) such that $X \doteq Z_1 \doteq Z_2 \cdots \doteq Z_n \doteq Y$. Define $\overset{*}{=}$ to be $\overset{\varepsilon}{=} \cup \doteq$. Then define $X \gtrdot t$, $t \in \Sigma$ if one of the following conditions hold:

a) $X \doteq Y$, $Y \in N$, $t \in \mathbf{L}_\varepsilon(Y)$;

b) $X \overset{\varepsilon}{=} t$;

c) $X \overset{\varepsilon}{=} Y$, $t \in \mathbf{L}_\varepsilon(Y) \cup \mathbf{L}(Y)$;

d) $Y \overset{*}{=} t$, $Y \in N$, $X \in \mathbf{R}_\varepsilon(Y) \cup \mathbf{R}(Y)$;

e) $Y_1 \overset{*}{=} Y_2$, $Y_1, Y_2 \in N$ $X \in \mathbf{R}_\varepsilon(Y_1) \cup \mathbf{R}(Y_1)$ $t \in \mathbf{L}_\varepsilon(Y_2) \cup \mathbf{L}(Y_2)$;

(4) Let $\$$, a symbol not in N∪Σ, be the left and right end-marker of sentential forms in G. Define:

$\$ \lessdot X$, for all X ∈ **L**($S$);

$\$ \gtrdot t$, for all t ∈ $\mathbf{L}_\varepsilon(S)$;

$X \gtrdot \$$, for all X ∈$\mathbf{R}_\varepsilon(S) \cup \mathbf{R}(S)$;

$\$ \gtrdot \$$, if S∈null(G).

Similar to Wirth-Weber relations, the purpose of the epsilon precedence relations is:

(*i*) to identify pairs of symbols which can legally appear in viable prefixes; and

(*ii*) to classify these pairs according to whether they occur at the left end of the handle ($\lessdot$), the right end of the handle ($\gtrdot$) or within the handle ($\doteq$) of right sentential forms.

Simple precedence grammars are uniquely invertible; that is, they do not have multiple productions with identical right parts. This property is used by simple precedence parsers at the time a reduction is called for; the stack is reduced by the unique production whose right part appears on top of the stack. Epsilon simple precedence grammars, however, are not necessarily uniquely invertible. This is due to the existence of ε-rules. A concept similar to unique invertibility is now defined for grammars with ε-rules.

DEFINITION 3.3 A CFG, G=(N,Σ,P,S), is said to be *Almost Uniquely Invertible* (AUI), if for all A, B $\in$ N, A$\neq$B, $A \to \beta \in$ P and B$\to\beta \in$ P imply $\beta=\varepsilon$.

3.1 Epsilon Simple Precedence Grammars

In this section, we introduce the class of epsilon simple precedence grammars.

DEFINITION 3.4 A reduced CFG, G=(N,Σ,P,S), is said to be an *Epsilon Simple Precedence* (ESP) grammar when

(1) G is almost uniquely invertible;

(2) G is cycle-free;

(3) epsilon precedence relations for G are pairwise disjoint;

(4) if A$\to\alpha$X and B$\to\varepsilon$ are two distinct productions in P, then (X,B) $\notin(\doteq \cup \lessdot)p$;

(5) for all Z', $Z'' \in N_\varepsilon$(G), $Z' \neq Z''$, $\rho(Z') \cap \rho(Z'')=\phi$ where for all Z $\in N_\varepsilon$(G), ρ(Z) is defined as follows:

$$\rho(Z)=\{(X,t) \mid (X,Z) \in (\doteq \cup \lessdot), (X,t) \in \gtrdot, (Z,t) \in (\doteq \cup \lessdot \cup \gtrdot)\};$$

(6) for all Z$\in N_\varepsilon$(G), (S,\$) $\notin\rho(Z)$.

An ESP grammar with i ε-rules is denoted ESP_i. A language defined by some ESP_i grammar is said to be an ESP_i language.

Conditions (1)-(6) of Definition 3.4 allow the implementation of a simple and deterministic bottom-up parser. Condition (3) guarantees that the next action of the parser can always be uniquely determined by examining the epsilon precedence relation that holds between the stack symbol and the next input symbol. Conditions (1), (4) and (5) simply state that if the next action of the parser is to "reduce," the production to be used in the reduction can be uniquely identified. Condition (6) guarantees that there will be no confusion as to whether "shift" or "accept." Finally, Condition (2) ensures that our epsilon precedence parsing algorithm (Algorithm 3.7) will terminate.

Before presenting the epsilon precedence parsing algorithm, we present two lemmas relating EP relations to derivations in a CFG. We establish that in a right sentential form, $\alpha\beta\omega$, with the handle of β, the relations $\doteq$, or $\lessdot$ hold between adjacent symbols of $\alpha\beta$ and the relation $\leftrightarrow$ holds between $SUFF_1(\alpha\beta)$ and $PREF_1(\omega)$. Moreover, we show that if ε is the handle of a right sentential form, $\alpha\omega$, derived form $\alpha Z\omega$, then $(SUFF_1(\alpha), PREF_1(\omega))$ belongs to $\rho(Z)$.

LEMMA 3.5 Let G=(N,Σ,P,S) be a proper CFG. If \$S\$ $\Rightarrow_{rm}^{*}$ $\alpha X\delta a\omega$ $\Rightarrow_{rm}^{+}$ $\alpha Xa\omega$, then $X \leftrightarrow a$.

Proof. $\alpha Xa\omega$ may be derived from \$S\$ in several ways. We consider each case separately.

CASE 1. \$S\$ $\Rightarrow_{rm}^{*}$ $\alpha X\delta a\omega \Rightarrow_{rm}^{+} \alpha Xa\omega$ = \$\$

In this case $S \Rightarrow^{+} \varepsilon$, and from Definition 3.4, we have \$ $\leftrightarrow$ \$.

CASE 2. S $\Rightarrow^{+}_{rm} \alpha' A z \Rightarrow_{rm} \alpha' \alpha'' X \delta' Y \beta z \Rightarrow^{*}_{rm} \alpha' \alpha'' X \delta' Y y z \Rightarrow^{*}_{rm}$

$\alpha' \alpha'' X \delta' \delta'' a x y z = \alpha X \delta a \omega \Rightarrow^{+}_{rm} \alpha X a \omega$

Here $A \to \alpha'' X \delta' Y \beta \in P$, and either $X \stackrel{+}{=} Y$ ($\delta' = \varepsilon$) and $a \subset \mathbf{L}_{\varepsilon}(Y)$, or $X \stackrel{\varepsilon}{=} Y$ ($\delta' \neq \varepsilon$) and $a \in \mathbf{L}(Y) \cup \mathbf{L}_{\varepsilon}(Y) \cup \{Y\}$. In either case it follows from Definition 3.4 that $X \Leftrightarrow a$.

CASE 3. S $\Rightarrow^{+}_{rm} \alpha' A z \Rightarrow_{rm} \alpha' \alpha'' B \delta' Y \beta z \Rightarrow^{*}_{rm} \alpha' \alpha'' B \delta' Y y z \Rightarrow^{*}_{rm}$

$\alpha' \alpha'' B \delta' \delta'' a x y z = \alpha' \alpha'' B \delta' \delta'' a \omega \Rightarrow^{*}_{rm} \alpha' \alpha'' B a \omega \Rightarrow^{*}_{rm}$

$\alpha' \alpha'' \vartheta X \lambda a \omega \Rightarrow^{*}_{rm} \alpha' \alpha'' \vartheta X a \omega = \alpha X a \omega$, where $\delta' \delta'' \lambda \neq \varepsilon$.

Since $\delta' \delta'' \lambda \neq \varepsilon$, at least one of λ, δ', or δ'' must be non-null.

SUBCASE 3.1 $\lambda \neq \varepsilon$

In this case $B \Rightarrow^{*}_{rm} \vartheta X \lambda \Rightarrow^{+}_{rm} \vartheta X$, and $X \in \mathbf{R}_{\varepsilon}(B)$. Also $Y \Rightarrow^{*}_{rm} \delta'' a x \Rightarrow^{*}_{rm} a x$, and we must have $a \in \mathbf{L}(Y) \cup \mathbf{L}_{\varepsilon}(Y) \cup \{Y\}$. Since $A \to \alpha'' B \delta' Y \beta \in P$, and $\delta' \Rightarrow^{*}_{rm} \varepsilon$, we have $B \stackrel{*}{=} Y$. Therefore $X \Leftrightarrow a$.

SUBCASE 3.2 $\delta' \neq \varepsilon$

Here, from the production $A \to \delta'' B \delta' Y \beta$, and derivation $\delta' \Rightarrow^{+}_{rm} \varepsilon$, we have $B \stackrel{\varepsilon}{=} Y$. Also derivations $B \Rightarrow^{*}_{rm} \vartheta X \lambda \Rightarrow^{*}_{rm} \vartheta X$, and $Y \Rightarrow^{*}_{rm} \delta'' a x \Rightarrow^{*}_{rm} a x$, imply $X \in \mathbf{R}(B) \cup \mathbf{R}_{\varepsilon}(B) \cup \{B\}$ and $a \in \mathbf{L}(Y) \cup \mathbf{L}_{\varepsilon}(Y) \cup \{Y\}$. It then follows from Definition 3.4 that $X \Leftrightarrow a$.

SUBCASE 3.3 $\delta'' \neq \varepsilon$

In this case, similar to subcase (3.1) we have $B \stackrel{*}{=} Y$. Also, derivations $B \Rightarrow^{*}_{rm} \vartheta X \lambda \Rightarrow^{*}_{rm} \vartheta X$ imply $X \in \mathbf{R}(B) \cup \mathbf{R}_{\varepsilon}(B) \cup \{B\}$. But the derivation $Y \Rightarrow^{*}_{rm} \delta'' a x \Rightarrow^{+}_{rm} a x$, implies $a \in \mathbf{L}_{\varepsilon}(Y)$, and thus $X \Leftrightarrow a$.

LEMMA 3.6 Let G=(N,Σ,P,S) be a proper CFG. If

$$\$S\$ ==>^{+}_{rm} X_p X_{p-1} \cdots X_{k+1} A \alpha_1 \cdots \alpha_q ==>$$
$$X_p X_{p-1} \cdots X_{k+1} X_k \cdots X_1 \alpha_1 \cdots \alpha_q ;$$

then

(1) For $p > i > k$, $(X_{i+1}, X_i) \in (\stackrel{\pm}{=} \cup \lessdot)$;

(2) $X_1 \doteq\!\!> \alpha_1$

(3) Either $k > 0$, and

(3.1) $X_{k+1} \lessdot X_k$;

(3.2) for $k > i \geq 1$, $X_{i+1} \stackrel{\pm}{=} X_i$;

or $k = 0$, and

(3.3) $(X_1, \alpha_1) \in \rho(A)$.

Proof. The proof is essentially the same as the proof given in [5] for grammars without ε-rules. We only observe that if $k=0$ (the handle is ε), conclusion (2) follows from Lemma 3.5 and conclusion (3.3) follows directly from conclusions (1) and (2) and Definition 3.4.

We now present the epsilon precedence parsing algorithm. This parsing algorithm resembles the traditional precedence parsers. Each time the relation that holds between the symbol on top of the top stack and the next input symbol is examined. If the relation $\stackrel{\pm}{=}$, or $\lessdot$ holds, then the next input symbol is shifted. If the relation $\doteq\!\!>$ holds, the stack is reduced by the production with the longest right part matching the string on top of the stack. If no such production with non-empty right part is found, then ρ of erasable symbols are examined and one ε-rule is selected to be used in reduction. The algorithm announces an error, if either no relation holds between the top stack and input symbol or no production can be selected for reduction. The correctness of Algorithm 3.7 is a direct result of Lemmas 3.5 and 3.6.

ALGORITHM 3.7 Shift-Reduce parsing algorithm for an ESP grammar.

INPUT: An ESP grammar, G=(N,Σ,P,S), in which the productions in P are numbered from 1 to p.

OUTPUT: A=(f,g), a pair of functions defining a shift-reduce parsing algorithm.

METHOD:

(1) Let $\$$ be the bottom marker for the stack and the right end-marker for the input.

(2) The shift-reduce function, f, is defined as:

a) f(\$S,\$) = accept;

b) f(X,t) = shift; for all $X \in V_G \cup \{\$\}$, $t \in \Sigma \cup \{\$\}$ such that $(X,t) \in (\lessdot \cup \doteq)$.

c) f(X,t) = reduce; for all $X \in V_G \cup \{\$\}$, $t \in \Sigma \cup \{\$\}$ such that $(X,t) \in \gtrdot$.

d) f(X,t) = error; otherwise.

(3) For $X \in V_G \cup \{\$\}$, $t \in \Sigma \cup \{\$\}$ and $\alpha \in \$V_G^*$ denoting the top l symbols of the parse stack, where l is the length of the longest right part, define the reduce function, g, as follows:

a) $g(\alpha,t) = i$, if $\alpha = \alpha'\beta$, $\beta \neq \varepsilon$, i: $B \rightarrow \beta$ and for all $A \rightarrow \delta\beta \in P$, $\delta\beta$ is not a suffix of α;

b) $g(\alpha,t) = i$, if $\alpha = \alpha' X$, $(X,t) \in \rho(Z)$, i: $Z \rightarrow \varepsilon$ and for all $A \rightarrow \beta \in P$, $\beta \neq \varepsilon$, β is not a suffix of α;

c) g(α,t) = error, otherwise.

We conclude this section by showing that the class of of ESP languages properly includes the class of simple precedence languages.

THEOREM 3.8 The class of simple precedence languages is properly included in the class of ESP languages.

Proof. Obviously every simple precedence grammar is also an ESP grammar. Hence the class of simple precedence languages is included in the class of ESP languages. To show the proper inclusion, consider the language $L_1=\{a\,0^n\,1^n \mid n>0\}\cup\{b\,0^n\,1^{2\times n} \mid n>0\}$, which is known not to be a simple precedence language [15]. Grammar G, shown below, is an ESP_1 grammar which defines L_1.

S → aX, S → bY, X → AX1,
X → A1, Y → BYC1, Y → BC1,
C → 1, A → Z0, B → 0,
Z → ε

Table 2 shows the sets **L**, $\mathbf{L_\varepsilon}$, **R**, and $\mathbf{R_\varepsilon}$ for the nonterminal symbols of G.

	L	L_ε	R	R_ε
S	a,b	ϕ	X,Y,1	ϕ
X	A,Z	0	1	ϕ
Y	B,0	ϕ	1	ϕ
A	Z	0	0	ϕ
B	0	ϕ	0	ϕ
C	1	ϕ	1	ϕ
Z	ϕ	ϕ	ϕ	ϕ

Table 2. **L**, $\mathbf{L_\varepsilon}$, **R**, $\mathbf{R_\varepsilon}$ for G.

	S	X	Y	A	B	C	Z	a	b	0	1	$
S												
X											$\doteq$	$\dot{>}$
Y						$\doteq$					$\dot{<}$	$\dot{>}$
A		$\doteq$		$\dot{<}$			$\dot{<}$			$\dot{>}$	$\doteq$	
B			$\doteq$		$\dot{<}$	$\doteq$				$\dot{<}$		
C											$\doteq$	
Z										$\doteq$		
a		$\doteq$		$\dot{<}$			$\dot{<}$			$\dot{>}$		
b			$\doteq$		$\dot{<}$					$\dot{<}$		
0										$\dot{>}$	$\dot{>}$	
1											$\dot{>}$	$\dot{>}$
$								$\dot{<}$	$\dot{<}$			

Table 3. The EP relations for G.

Clearly G is cycle-free and AUI. Moreover, the EP relations for G, shown in Table 3, are pairwise disjoint. Also, G has only one ε-rule and condition (5) of Definition 3.4 is satisfied. Additionally, the rightmost symbol of no production in G is related to Z, and $(S,\$) \notin \rho(Z)$. Hence, G is ESP and therefore L_1 is an ESP language.

3.2 Epsilon Weak Precedence Grammars

Algorithm 3.7 shifts the next input symbol if the relation between the top stack symbol is either $\doteq$ or $\dot{<}$. Therefore, in a manner similar to how simple precedence grammars were generalized to weak precedence grammars by allowing the relations $\doteq$ and $\dot{<}$ to intersect, we generalize epsilon simple precedence grammars. The resulting class of grammars is called epsilon weak precedence grammars.

DEFINITION 3.9 A reduced CFG G=(N,Σ,P,S) is said to be an *Epsilon Weak Precedence* (EWP) grammar if:

(1) G is almost uniquely invertible;

(2) G is cycle-free;

(3) $(\stackrel{\pm}{=} \cup \lessdot) \cap \gtrdot = \phi$;

(4) if A $\rightarrow \alpha X \beta$ and $B \rightarrow \beta$ are two distinct productions in P, $\beta \in V_G^*$, then (X,B) $\notin (\stackrel{\pm}{=} \cup \lessdot)$;

(5) for all Z', $Z'' \in N_\varepsilon$(G), $Z' \neq Z''$, $\rho(Z') \cap \rho(Z'') = \phi$ where for all Z $\in N_\varepsilon$(G), ρ(Z) is defined as follows:

$$\rho(Z) = \{(X,t) \mid (X,Z) \in (\stackrel{\pm}{=} \cup \lessdot),\ (X,t) \in \gtrdot,\ (Z,t) \in (\stackrel{\pm}{=} \cup \lessdot \cup \gtrdot)\};$$

(6) for all $Z \in N_\varepsilon$(G), (S,\$) $\notin \rho(Z)$;

An EWP grammar with i ε-rules is denoted EWP_i. A language defined by some EWP_i grammar is said to be an EWP_i language.

Although the class of EWP grammars is more general than the class of simple precedence grammars, the class of languages defined is identical to that defined by epsilon simple precedence grammars. To establish the equivalence of ESP_i and EWP_i grammars, we present an algorithm to convert arbitrary EWP_i grammars to equivalent ESP_i grammars.

ALGORITHM 3.10 Conversion of an arbitrary EWP_i grammar to an equivalent ESP_i grammar.

INPUT: G=(N,Σ,P,S), an arbitrary EWP_i grammar.

OUTPUT: $G'=(N',\Sigma,P',S)$, an equivalent ESP_i grammar.

METHOD:

(1) $P'=P$

$N'=N$

(2) While there exists a pair of symbols X and Y in $V_{G'}$ such that $X \stackrel{+}{=} Y$, and $X \not\Leftarrow Y$ do

Let β be a string in $V_{G'}^*$ such that there exists a production $A \rightarrow \alpha XY\beta$ in P', $\alpha \in V_{G'}^*$.

Let $[Y\beta]$ be a new symbol not found in N'

(2.1) Add $[Y\beta]$ to N'

(2.2) Add $[Y\beta] \rightarrow Y\beta$ to P'

(2.3) Replace each production $A \rightarrow \delta XY\beta$ in P' by $A \rightarrow \delta X[Y\beta]$

(2.4) If $B \rightarrow Y\beta$ is a production in P', then replace $B \rightarrow Y\beta$ by $B \rightarrow [Y\beta]$

In our next two lemmas, we prove that the grammar G', produced by Algorithm 3.10 is an ESP_i grammar and L(G)=L(G').

LEMMA 3.11 In Algorithm 3.10, L(G)=L(G').

Proof. Clearly, the first time step (2) of the algorithm is reached, we have that G=G', and therefore, L(G)=L(G'). We show that applying steps (2.1) - (2.4) does not change L(G'). Suppose $A \rightarrow \alpha XY\beta$ is the production selected in step (2). Each production $A \rightarrow \delta XY\beta$ is replaced by the production $A \rightarrow \delta X[Y\beta]$ in step (2.3). Moreover, the production $[Y\beta] \rightarrow Y\beta$ is added in step (2.2). Hence, after applying

step (2), for all $\delta \in V_{G'}^{*}$ the derivation A==>$\delta XY\beta$, is replaced by A==>$\delta X[Y\beta]$==>$\delta XY\beta$. Additionally, if B→$Y\beta$ is a production in P', then it is replaced by the production B→$[Y\beta]$ in step (2.4). Thus, the derivation B==>$Y\beta$ is replaced by B==>$[Y\beta]$==>$Y\beta$. Observing that $[Y\beta]$ is a new symbol which derives only $Y\beta$, and appears only in the right parts of the productions added in steps (2.3) and (2.4), we conclude that L(G)=L(G').

LEMMA 3.12 In Algorithm 3.10, G' is ESP_i.

Proof. Clearly, all of the productions added to G' in steps (2.2) - (2.4) have non-empty right parts. Moreover, no ε-rule is removed from G' in steps (2.3) and (2.4). Hence, the number of ε-rules in G' is the same as the number of ε-rules in G. We show that G' is ESP.

1. G' is cycle-free.

An argument similar to the one provided for Lemma (3.11), proves that applying steps (2.1) and (2.4) does not introduce a cycle to G'. Hence, if G' has a cycle, then G must have a cycle. The assumption that G is EWP, implies G' is cycle-free.

2. G' is almost uniquely invertible.

Each time steps (2.2) - (2.4) are applied to G', productions of the form A→$\delta X[Y\beta]$, B→$[Y\beta]$ and $[Y\beta]$→$Y\beta$ are added to P'. Obviously, $[Y\beta]$ is a new symbol added to N' in step (2.1) and P' does not have any production with a right part of $\delta X[Y\beta]$, or $[Y\beta]$. Moreover, G' is almost uniquely invertible and can have at most one production with the right part of $Y\beta$. If any such production exists, it is removed from P' in step (2.4) of the algorithm. Hence, applying steps

(2.2) - (2.4) of the algorithm adds productions with unique right parts to G'. Considering that G is almost uniquely invertible, we conclude that G' is almost uniquely invertible.

3. If A $\to \alpha B$ and X $\to \varepsilon$ are productions in P', then $(B,X) \notin (\doteq \cup \lessdot)$.

Assume the contrary. Then the relation (B,X) must also be a relation in G. This is due to the fact that the added relations do not involve any symbol in $N_\varepsilon(G)$. Hence, B and X are symbols in V_G, and it follows that A$\to\alpha B$ and X$\to\varepsilon$ belong to P, and we have a contradiction for G being EWP.

4. Epsilon Precedence relations are pairwise disjoint for G'.

Obviously, each time steps (2.2) - (2.4) of Algorithm 3.10 are applied to G', the following relations are added to G':

(i) X$\doteq [Y\beta]$,

(ii) $[Y\beta] \lessdot t$,

where A$\to\delta XY\beta$ is the production selected in step (2) and $(A,t) \in (\doteq \cup \lessdot \cup \gtrdot)$ in G'. Moreover, if a production B$\to Y\beta$ is selected in step (2.4), the following relations are added to G'.

(iii) $[Y\beta] \gtrdot t$, where $(B,t) \in (\doteq \cup \lessdot \cup \gtrdot)$ in G'

(iv) $Z \lessdot [Y\beta]$, where $(Z,B) \in (\doteq \cup \lessdot)$ in G'

Clearly, since $[Y\beta]$ is a new symbol, the relations described in (i) - (iv) cannot conflict with any existing relation. Additionally, relations $X \doteq [Y\beta]$ and $Z \lessdot [Y\beta]$ do not conflict. Otherwise, we must have that X=Z, and $(X,B) \in (\doteq \cup \lessdot)$; but A$\to\alpha XY\beta$ and B$\to Y\beta$ are productions in G', and it follows that $(X,B) \notin (\doteq \cup \lessdot)$. Each time step (2) of the algorithm is performed, one of the productions that give rise to the relation X$\doteq$Y is removed. The algorithm terminates

when no such pair of symbols can be found. Hence, the EP relations for G' are pairwise disjoint.

5. For all $Z, Z' \in N_\varepsilon(G'), Z \neq Z', \rho(Z) \cap \rho(Z') = \phi$

It is clear from the argument for case (4) that the relations added to G' do not involve the symbols in N_ε(G). Hence, for all $Z \in N_\varepsilon$(G), ρ(Z) in G is identical to ρ(Z) in G'. But, G is EWP and for all $Z, Z' \in N_\varepsilon$(G), $Z \neq Z'$, $\rho(Z) \cap \rho(Z') = \phi$. Hence, the same property holds for the members of N_ε in G'.

6. For all $Z \in N_\varepsilon$(G), (S,\$) $\notin \rho(Z)$;

An argument similar to the one provided for case (5) will establish this result.

3.3 Extended Epsilon Weak Precedence Grammars

We conclude this chapter by presenting the class of extended epsilon weak precedence grammars. This generalization is similar to the generalization of weak precedence grammars to extended weak precedence grammars [44], except here ε-rules are allowed. First we define the extended epsilon precedence relations.

DEFINITION 3.13 Let G=(N,Σ,P,S) be a CFG. We define the (m,n) *Epsilon Precedence* relations, $\lessdot$, $\stackrel{\pm}{=}$, and $\gtrdot$ on

$$(V_G \cup \{\$\}^m) \times (V_G \cup \{\$\}^n)$$

as follows. Let $\$^m S \$^n ==>^*_{rm} \alpha A \omega ==> \alpha X_1 X_2 \cdots X_l \omega$, $l \geq 0$ be a rightmost derivation in G. Then,

(1) If $l > 0$, then for all i, $1 \leq i < l$, $SUFF_m(\alpha X_1 X_2 \cdots X_i) \stackrel{\pm}{=} \beta$, where either

$\beta = PREF_n(X_{i+1} X_{i+2} \cdots X_l \omega)$; or

$X_{i+1} \in \Sigma$, and $\beta \in FIRST_n(X_{i+1} X_{i+2} \cdots X_l \omega)$.

(2) If $\iota > 0$, then $SUFF_m(\alpha) \lessdot \beta$, where either

$$\beta = PREF_n(X_1 X_2 \cdots X_l \omega);$$ or

$$X_1 \subset \Sigma, \text{ and } \beta \subset FIRST_n(X_1 X_2 \cdots X_l \omega).$$

(3) $SUFF_m(\alpha X_1 X_2 \cdots X_l) \doteq PREF_n(\omega)$

DEFINITION 3.14 A reduced CFG G=(N,Σ,P,S) is said to be an (m,n,p,q)EWP grammar if:

(1) G is almost uniquely invertible;

(2) G is cycle-free;

(3) The extended (m,n)epsilon precedence relation $\doteq$ is disjoint from the union of the extended (m,n) epsilon precedence relations $\lessdot$ and $\stackrel{\pm}{=}$.

(4) If $X \to \alpha\beta$ and $Y \to \beta$ are distinct productions, $\alpha\beta \neq \varepsilon$, then for no $\gamma, \delta \in (V_G \cup \{\$\})^*$ is it true that $(SUFF_m(\gamma\alpha), PREF_n(Y\delta)) \in (\lessdot \cup \stackrel{\pm}{=})$;

(5) For all Z', $Z'' \in N_\varepsilon(G)$, $Z' \neq Z''$, $\rho_{p,q}(Z') \cap \rho_{p,q}(Z'') = \phi$ where for all $Z \in N_\varepsilon(G)$, $\rho_{p,q}(Z)$ is defined as follows:

$$\rho(Z) = \{(SUFF_p(\alpha), PREF_q(\omega)) \mid \$^p S \$^q \Longrightarrow_{rm}^* \alpha Z \omega \text{ is a derivation in G}\}$$

A (1,1,p,q)EWP grammar is denoted (p,q)EWP.

CHAPTER 4

THE TRANSFORMATION OF LR(1) GRAMMARS INTO EWP GRAMMARS

The use of epsilon simple and weak precedence grammars as the basis for implementing practical parsers suffers two major drawbacks:

(1) The classes of epsilon simple and weak precedence grammars have a small intersection with the class of grammars one would naturally write for a programming language. That is, usually, a natural grammar for a given language is not epsilon precedence.

(2) Epsilon precedence parsers do not in general have the viable prefix property. Consequently, these parsers do not usually detect syntax errors present in the input at the earliest possible time during a parse.

Automatic conversion of arbitrary context-free grammars to equivalent EWP grammars is not possible since, as we will show in Chapter 6, EWP grammars define exactly the deterministic languages and the problem of deciding whether or not an arbitrary context-free grammar defines a deterministic language is known to be unsolvable [5,22]. Therefore, to overcome the first problem mentioned above, we have designed algorithms to automatically convert LR(1) grammars to equivalent (2,1)EWP and (1,1)EWP grammars. These algorithms accept an LR(1) grammar, G, as input and produce an equivalent EWP grammar, G', by encoding the entire GOTO graph of G's LR(1) parser into special erasing nonterminals.

To achieve our second goal, we have defined the class of Viable Prefix EWP grammars and show that the EWP parser for any Viable Prefix EWP grammar has the viable prefix property. Moreover, we establish that every deterministic context-free language is defined by some Viable Prefix EWP grammar by showing that the EWP grammars obtained by applying our algorithm to arbitrary LR(1) grammars are Viable Prefix EWP grammars.

4.1 Conversion of LR(1) Grammars to Equivalent (2,1)EWP Grammars

In this section we present an algorithm to convert arbitrary LR(1) grammars to equivalent (2,1)EWP grammars.

ALGORITHM 4.1 Conversion of arbitrary LR(1) grammars to equivalent (2,1)EWP grammars.

INPUT: G=(N,Σ,P,S), an arbitrary LR(1) grammar.

OUTPUT: $G'=(N',\Sigma,P',S')$, an equivalent (2,1)EWP grammar.

METHOD: Let **C** be the canonical collection of LR(1) states for G, such that the states in **C** are labeled $I_0, I_1, \cdots, I_m$, where I_0 is the initial state in **C**, and for all i, $0 \le i \le m$, $I_i \notin V_G$.

(1) $S' = [I_0, S]$;

$N_1 = \{ [I,A] \mid$ there exists an item $[A \rightarrow .\alpha \mid u]$ in I, for some $I \in \mathbf{C}\}$

$\cup \{I \mid I \in \mathbf{C}\}$;

$P_1 = \{I \rightarrow \varepsilon \mid I \in \mathbf{C}\}$.

(2) For each state $I \in \mathbf{C}$, and each item $[A \rightarrow .A_1 A_2 \cdots A_n \mid u] \in I$, $n \ge 0$, do

(2.1) Let K=[GOTO($I, A_1 A_2 \cdots A_n$),Z], where

Z=$\{v \mid [A \rightarrow A_1 A_2 \cdots A_n . \mid v] \in$ GOTO($I, A_1 A_2 \cdots A_n$)$\}$.

Let Y_0 be defined as follows:

$$Y_0 = \begin{cases} \varepsilon & \text{if } n>0, \text{ and } A_1 \in N; \\ I & otherwise; \end{cases}$$

For each i, $1 \le i \le n$, define:

$$X_i = \begin{cases} [GOTO(I, A_1 A_2 \cdots A_{i-1}), A_i] & A_i \in N; \\ A_i & A_i \in \Sigma; \end{cases}$$

For each i, $1 \le i < n$, define:

$$Y_i = \begin{cases} GOTO(I, A_1 A_2 \cdots A_i) & A_{i+1} \in \Sigma; \\ \varepsilon & A_{i+1} \in N; \end{cases}$$

(2.2) Add $[I,\text{A}] \rightarrow Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{n-1} Y_{n-1} X_n K$ to P_1;

Add $K \rightarrow \varepsilon$ to P_1;

Add K to N_1.

(3) Construct G' by reducing $G_1 = (N_1, \Sigma, P_1, S')$

It is easy to observe that N' can be written as $N'' \cup N_r \cup N_s$, where N'', N_r, and N_s are disjoint sets defined as follows:

N'' = {[I,A] | I∈**C**, A∈N, $[I,\text{A}] \rightarrow \alpha \in P'$, for some $\alpha \in V_G^+$}

N_r = {[k,Z] | k ∈**C**, $Z \subseteq \Sigma \cup \{\varepsilon\}$}.

N_s = { I | I ∈**C**}.

Also, each production in P' is either of the form A→ε, for some A∈$N_s \cup N_r$, or $[I,\text{A}] \rightarrow Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{n-1} Y_{n-1} X_n$ K, for some $n \ge 0$, where

(a) K=[k,Z]∈N_r.

(b) For each i, $1 \le i \le n$, either $X_i \in N''$ and $Y_{i-1} = \varepsilon$, or $X_i \in \Sigma$ and $Y_{i-1} \in N_s$.

Furthermore, for each production

$$[\text{I,A}] \rightarrow Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{n-1} Y_{n-1} X_n [k,Z],\ n \ge 0,$$

satisfying properties (a) - (b) above, there exists a production $A \rightarrow B_1 B_2 \cdots B_n$ in P such that

(c) For all i, $1 \leq i \leq n$,

$$B_i = \begin{cases} X_i & X_i \in \Sigma; \\ A_i & X_i = [I_i, A_i],\ I_i \in \mathbf{C},\ A_i \in N. \end{cases}$$

(d) There exists $v \in Z$, such that the LR(1) item $[A \rightarrow . B_1 B_2 \cdots B_n \mid v]$ is in the state I of **C**.

(e) For all $v \in Z$, the LR(1) item $[A \rightarrow B_1 B_2 \cdots B_n . \mid v]$ belongs to the state k =GOTO($I, B_1 B_2 \cdots B_n$) of **C**.

(f) For all i, $1 \leq i \leq n$, such that $X_i = B_i \in \Sigma$, there exists $v \in Z$ such that the LR(1) item $[A \rightarrow B_1 B_2 \cdots B_{i-1} . B_i \cdots B_n \mid v]$ belongs to the state Y_{i-1} of **C**.

(g) For all i, $1 \leq i \leq n$, such that $X_i = [I_i, B_i] \in N''$, there exists $v \in Z$ such that $[A \rightarrow B_1 B_2 \cdots B_{i-1} . B_i \cdots B_n \mid v]$ belongs to the state I_i =GOTO($I, B_1 B_2 \cdots B_{i-1}$) of **C**.

Before proving that G' of Algorithm 4.1 is (2,1)EWP and L(G') = L(G), we show how derivations in G' relate to those in G.

LEMMA 4.2 Let $G' = (N', \Sigma, P', S')$ be the (2,1)EWP grammar constructed from an arbitrary LR(1) grammar, G=(N,Σ,P,S), by Algorithm 4.1. Let $[I, A] ==>^{*}_{rm} \alpha$, $[I, A] \in N''$, $\alpha \in V_{G'}^{*}$, be a derivation in G'. Define the homomorphism h as follows:

$$h(C) = \begin{cases} C & C \in \Sigma; \\ B & C = [J, B],\ J \in \mathbf{C},\ B \in N; \\ \varepsilon & C \rightarrow \varepsilon \in P'. \end{cases}$$

Then, $A ==>^{*}_{rm} h(\alpha)$ is a derivation in G.

Proof. The derivation $[I,\mathrm{A}] ==>^{*}_{rm} \omega$ in G' can be written as

$$[I,\mathrm{A}] = \alpha_0 ==>_{rm} \alpha_1 ==>_{rm} \alpha_2 ==>_{rm} \cdots ==>_{rm} \alpha_n.$$

We prove by induction that for each k, $0 \leq k \leq n$, $\mathrm{A} ==>^{*}_{G} h(\alpha_k)$ is a derivation in G.

BASIS: $k=0$. For $k=0$, we have that $\alpha_0=[I,\mathrm{A}]$ and $h(\alpha_0)=\mathrm{A}$, and $\mathrm{A} ==>^{*} A$ is a derivation of length zero in G.

INDUCTIVE STEP: Assume that for all k, $0 \leq k < n$, $n \geq 0$, $\mathrm{A} ==>^{*}_{rm} h(\alpha_k)$ is a derivation in G, and consider the derivation $[I,\mathrm{A}] ==>^{*}_{rm} \alpha_{n-1} ==> \alpha_n$. The production used for deriving α_n from α_{n-1} may either be an epsilon production, or a non-epsilon production. Clearly, in the first case $h(\alpha_{n-1}) = h(\alpha_n)$ and we have that $h(\alpha_n) ==>^{*}_{G} h(\alpha_{n+1})$. In the latter case, however, the production used must be of the form $[I,B] \rightarrow Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{l-1} Y_{l-1} X_l \mathrm{K}$, for some $l \geq 0$, where $\alpha_{n-1} = \alpha'_{n-1}[I,B]\omega$, $\omega \in \Sigma^{*}$. But it follows from the form of the non-epsilon productions in G' that $\mathrm{K} \rightarrow \varepsilon \in P'$, and for all i, $0 \leq i < l$, $Y_i ==>^{*1}_{G} \varepsilon$, and there exists a production $\mathrm{B} \rightarrow B_1 B_2 \cdots B_l$ in P, such that for all i, $0 < i \leq l$, either $X_i = B_i$ is a symbol in Σ; or X_i is a symbol of the form $[I_i,B_i]$, for some $I_i \in \mathbf{C}$, and $B_i \in N$. Therefore, $h(Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{l-1} Y_{l-1} X_l K) = B_1 B_2 \cdots B_l$. Hence,

$$h(\alpha_{n-1}) = h(\alpha'_{n-1}[I,B]\omega) = h(\alpha'_{n-1}) h([I,B]) h(\omega) = h(\alpha'_{n-1}) B \omega, \text{ and}$$

$$h(\alpha_n) = h(\alpha'_{n-1})\, h(Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{l-1} Y_{l-1} X_l K)\, h(\omega) =$$

$$h(\alpha'_{n-1}) B_1 B_2 \cdots B_l \omega,$$

where $\mathrm{B} \rightarrow B_1 B_2 \cdots B_l$ is a production in P. Therefore, in either case $h(\alpha_{n-1}) ==>^{*}_{rm} h(\alpha_n)$ is a derivation in G, and it follows from the inductive hypothesis that $\mathrm{A} ==>^{*} h(\alpha_n)$ is a derivation in G.

We now show that G and G' are equivalent.

LEMMA 4.3 In Algorithm 4.1, L(G)=L(G').

Proof. We prove that any arbitrary nonterminal of the form $[I,A]\in N'$ derives in G' exactly those terminal strings that are generated by $A\in N$ in G. That is,

$$(1)\quad [I,A] \Longrightarrow_{G'}^{*} \omega \quad \text{if and only if} \quad A \Longrightarrow_{G}^{*} \omega, \quad \omega\in\Sigma^{*}.$$

If: Suppose $A \Longrightarrow_{G}^{n} \omega$, we prove by induction on n that for all $I\in\mathbf{C}$ such that $[I,A]\in N'$, we have that $[I,A]\Longrightarrow_{G'}^{*} \omega$.

BASIS: $n=1$. For $n=1$, $A\to\omega$ is a production in P. Clearly, if $\omega=\varepsilon$, then P' has a production $[I,A]\to I$K where $I\to\varepsilon$ and K$\to\varepsilon$ belong to P' and therefore, $[I,A]\Longrightarrow^{*}\varepsilon$ is a derivation in G'. Let $\omega=\omega_1\omega_2\cdots\omega_m$, $m\geq 0$, where for all i, $1\leq i\leq m$, $\omega_i\in\Sigma$. It then follows from the construction that for all $I\in\mathbf{C}$ such that $[I,A]\in N'$, there exist $Y_1, Y_2, \cdots Y_{m-1}$ and K in $N_{\varepsilon}(G')$ such that $[I,A]\to I\omega_1 Y_1\omega_2 Y_2\cdots Y_{m-1}\omega_m$K is a production in P'. Thus the following derivation exists in G':

$$\begin{aligned}
[I,A] &\Longrightarrow I\omega_1 Y_1\omega_2 Y_2\cdots Y_{m-1}\omega_m \text{K} \\
&\Longrightarrow I\omega_1 Y_1\omega_2 Y_2\cdots Y_{m-1}\omega_m \\
&\Longrightarrow I\omega_1 Y_1\omega_2 Y_2\cdots Y_{m-2}\omega_{m-1}\omega_m \\
&\Longrightarrow^{*} I\omega_1\omega_2\cdots\omega_{m-1}\omega_m \\
&\Longrightarrow \omega_1\omega_2\cdots\omega_{m-1}\omega_m = \omega,
\end{aligned}$$

and we have that $[I,A] \Longrightarrow_{G'}^{*} \omega$.

INDUCTIVE STEP: Assume that for all k, $k<n$, $n>1$, $A\Longrightarrow_{G}^{k} \omega$, implies for all I in **C** such that $[I,A]\in N'$, we have that $[I,A]\Longrightarrow_{G'}^{*} \omega$. Now consider a derivation of the form $A\Longrightarrow_{G}^{n} \omega$. Since $n>1$, $A\Longrightarrow_{G}^{n} \omega$ may be written as

$$A\Longrightarrow_{G} A_1A_2\cdots A_m \Longrightarrow_{G}^{n-1} \omega_1\omega_2\cdots\omega_m = \omega.$$

That is, $A \rightarrow A_1 A_2 \cdots A_m$, $m>0$, is a production in P, and for all i, $1 \leq i \leq m$, $A_i ==>_G^{n_i} \omega_i$, $n_i < n$, $\omega_i \in \Sigma^*$. Therefore, it follows from the construction that P' contains a production of the form

$$[I,A] \rightarrow Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{m-1} Y_{m-1} X_m K$$

such that $K \rightarrow \varepsilon \in P$, $Y_i ==>^{*1} \varepsilon$, $0 \leq i < m$, and for all i, $0 < i \leq m$ either $X_i = A_i \in \Sigma$, or $A_i \in N$ and $X_i = [I_i, A_i] \in N''$ for some $I_i \in \mathbf{C}$. If $X_i \in \Sigma$, we immediately conclude that $n_i = 0$ and $A_i = X_i = \omega_i$. If $X_i \in N''$, then from the inductive hypothesis, we have that $A_i ==>_G^{n_i} \omega_i$, $n_i < n$, implies that $X_i = [I_i, A_i] ==>_{G'}^{*} \omega_i$. Thus the following derivation exists in G'.

$$\begin{aligned}
[I,A] &==> Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{m-1} Y_{m-1} X_m K \\
&==> Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{m-1} Y_{m-1} X_m \\
&==>^{*} Y_0 X_1 Y_1 X_2 Y_2 \cdots Y_{m-1} \omega_m \\
&==>^{*} Y_0 X_1 Y_1 X_2 Y_2 \cdots Y_{m-2} \omega_{m-1} \omega_m \\
&==>^{*} Y_0 \omega_1 \omega_2 \cdots \omega_{m-1} \omega_m \\
&==>^{*} \omega_1 \omega_2 \cdots \omega_{m-1} \omega_m = \omega,
\end{aligned}$$

The above derivation may be written as $[I,A] ==>_{G'}^{*} \omega$ and the proof is complete.

Only if: Suppose $[I,A] ==>_{G'}^{*} \omega$. It then follows from Lemma 4.2 that $A ==>^{*} h(\omega)$ is a derivation in G. But $\omega \in \Sigma^*$ and therefore, $h(\omega) = \omega$. Hence $A ==>_G^{*} \omega$.

The special case of (1) where I is the label of the initial state in the canonical collection of LR(1) items for G and A=S results in the statement

$$S' ==>_{G'}^{*} \omega \quad \text{if and only if} \quad S ==>_G^{*} \omega, \quad \omega \in \Sigma^*$$

which proves that L(G)=L(G').

We now prove that G' of Algorithm 4.1 is (2,1)EWP.

LEMMA 4.4 In Algorithm 4.1, G' is a (2,1)EWP grammar.

Proof. We prove that G' is a (2,1)EWP grammar by showing that G' satisfies each of the conditions of Definition 3.14.

1. G' is Almost Uniquely Invertible.

If G' is not Almost Uniquely Invertible, there must exist two distinct productions $A' \rightarrow \alpha$ and $B' \rightarrow \alpha$, $\alpha \neq \varepsilon$ in P'. But $\alpha \neq \varepsilon$ implies that $\alpha = Y_0 \alpha'[k,Z]$, for some $\alpha' \in V_{G'}^*$, $Y_0 \in N_s \cup \{\varepsilon\}$, and $[k,Z] \in N_T$. Let $A' = [I,A]$ and $B' = [J,B]$, for some A, B$\in N$, and $I, J \in$ **C**. Then, P must contain productions $A \rightarrow A_1 A_2 \cdots A_n$, and $B \rightarrow A_1 A_2 \cdots A_n$, $n \geq 0$, such that for all $v \in Z$ the items $[A \rightarrow A_1 A_2 \cdots A_n . \mid v]$ and $[B \rightarrow A_1 A_2 \cdots A_n . \mid v]$ both belong to the state k of **C**. But G is assumed to be LR(1); therefore, these two items cannot be distinct. Otherwise, state k of **C** has a reduce/reduce conflict and G is not LR(1). Thus A = B. Now consider I and J. If $Y_0 \neq \varepsilon$, then we must have that $I = J = Y_0$. On the other hand, if $Y_0 = \varepsilon$, then $PREF_1(\alpha)$ is a nonterminal of the form $[Y', A_1]$. This time, we must have that $I = J = Y'$. Hence, $A' = B'$, and G' is Almost Uniquely Invertible.

2. G' is cycle-free.

Assume the contrary and let $[I,A] ==>^+ [I,A]$ be a derivation in G'. Applying Lemma 4.2 to this derivation, we must have that $A ==>^+ A$ is a derivation in G. Hence, if G' has a cycle, then G must have a cycle. But G is assumed to be LR(1) and therefore, cycle-free. Thus, G' is cycle-free.

3. In G', (1,1) EP relation $\doteq>$ is disjoint from $(\doteq \cup \lessdot)$.

If $(\doteq \cup \lessdot) \cap \doteq> \neq \phi$, for G', then there must exist C and D in $V_{G'}$, such that either $(C,D) \in (\doteq \cap \doteq>)$, or $(C,D) \in (\lessdot \cap \doteq>)$. We consider each potential conflict

of EP relations separately and in each case show that the conflict may not be present in G'.

CASE 1. C$\lessdot$ D and C $\doteq$ D.

If C$\lessdot$ D, then there must exist a production A$\rightarrow\alpha CX\beta$ in P' such that X$\in N'$ and D$\in$**L**(X). But from the construction we have that for all X$\in N'$ and $a\in\Sigma$, a does not belong to **L**(X). Therefore, D is not a terminal symbol. But C $\doteq$ D can hold only if D$\in\Sigma$. Thus, ($\lessdot\cup\doteq$)$=\phi$ for G'.

CASE 2. C$\stackrel{\pm}{=}$ D and C $\doteq$ D

Since C $\doteq$ D, we must have that D$\in\Sigma$. Also, if C$\stackrel{\pm}{=}$ D, there must exist a production A$\rightarrow\alpha CD\beta$ in P'. But D$\in\Sigma$, and it follows from the form of the productions in P' that $C\in N_s$. Therefore, A$\rightarrow\alpha CD\beta$ is of the form $[I,A']\rightarrow Y_0X_1Y_1X_2Y_2\cdots X_{n-1}Y_{n-1}X_n[k,Z]$, $n\geq1$, where C=Y_{i-1} and D=X_i, $1\leq i\leq n$. Thus, for some $v\in Z$ there must exist an LR(1) item, $[A'\rightarrow B_1\cdots B_{i-1}.B_i\cdots B_n\mid v]$ in state C of **C** such that $B_i=X_i$=D. Moreover, C is a symbol in N_s and it follows from the form of the productions in G' that if M$\rightarrow\alpha I\beta$, $I\in$**C**, is a production in P', then either $PREF_1(\beta)\in\Sigma$, or $\alpha=\varepsilon$ and β is a symbol N_r. Therefore, for C $\doteq$ D to hold, we must have a production

$$[I',E]\rightarrow Y'_0X'_1Y'_1X'_2Y'_2\cdots X'_{m-1}Y'_{m-1}X'_m[k',Z'],\ m\geq2,$$

in P' such that for some l and j, $1\leq l<j\leq m$, we have that

$$X'_l==>^*\alpha[I'',F]==>\alpha I''[k'',Z''],$$
$$Y'_lX'_{l+1}Y'_{l+1}\cdots X'_{j-1}Y'_{j-1}==>^*\varepsilon$$
$$D\in L_\varepsilon(X'_j)\cup\{X'_j\}$$

where C=I''. That is, there must exist a production $F'\rightarrow\varepsilon$ in P such that [F$\rightarrow\varepsilon.\mid D$] is an LR(1) item in state C of **C**. This item along with the item

$[A \rightarrow B_1 \cdots B_{i-1}.DB_{i+1} \cdots B_n \mid v]$ can belong to state C of **C** if G is not LR(1). But G is LR(1) and therefore, for all $C \in N_s$ and $D \in \Sigma$, both $C \stackrel{+}{=} D$ and $C \Leftrightarrow D$ cannot hold together in G'.

4. If $A \rightarrow \alpha X \beta$, and $B \rightarrow \beta$ are productions in P', then $(X,B) \notin (\stackrel{+}{=} \cup \Leftarrow)$.

We consider two cases:

CASE 1. $\beta = \varepsilon$.

Clearly, if $A \rightarrow \alpha X \in P'$, then $X \in N_T$. But the members of N_T appear only as the rightmost symbol in the right-part of any production and P' does not have any singleton production. Thus, if $X \in N_T$ and $Y \rightarrow \delta_1 X \delta_2 \in P'$, then we must have that $\delta_1 \neq \varepsilon$ and $\delta_2 = \varepsilon$. Hence, for all $C \in N'$, $X \in N_T$, $X \notin \mathbf{L}(C)$. Therefore, for all $B \in V_{G'}$, $X \in N_T$, $(X,B) \notin (\stackrel{+}{=} \cup \Leftarrow)$.

CASE 2. $\beta \neq \varepsilon$.

To establish our result for the case $\beta \neq \varepsilon$, we show that if $A \rightarrow \alpha X \beta$ is a production in P', $\alpha \in V_{G'}^*$, and $X \in V_{G'}$, then $B \rightarrow \beta$ is not a production in P' for any $B \in N'$.

Suppose for the sake of contradiction that $A \rightarrow \alpha X \beta$ and $B \rightarrow \beta$, $\beta \neq \varepsilon$, are productions in P'. Let $\beta = \beta'[k, Z]$, $k \in \mathbf{C}$, $Z \subseteq \Sigma \cup \{\varepsilon\}$. Then, there must exist two productions $A' \rightarrow A_1 A_2 \cdots A_m$ and $B' \rightarrow A_j A_{j+1} \cdots A_m$, $m \geq 1$, $1 \leq j \leq m$, such that for all $u \in Z$, the items $[A' \rightarrow A_1 A_2 \cdots A_m . \mid u]$, and $[B' \rightarrow A_j A_{j+1} \cdots A_m . \mid u]$ belong to the state k of **C**. But G is assumed to be LR(1) and these two items cannot be distinct. Hence G' satisfies condition (3) of Definition 3.14.

5. For all $Z_1, Z_2 \in N_\varepsilon(G')$, $Z_1 \neq Z_2$, $\rho_{2,1}(Z_1) \cap \rho_{2,1}(Z_2) = \phi$.

Suppose not and there exist Z_1 and Z_2, $Z_1 \neq Z_2$, $Z_1, Z_2 \in N_\varepsilon(G')$, such that $\rho_{2,1}(Z_1) \cap \rho_{2,1}(Z_2) \neq \phi$; that is, there exists (AB,a) in $\rho_{2,1}(Z_1) \cap \rho_{2,1}(Z_2)$, for some A, B in $V_{G'}$, and $a \in \Sigma$. Clearly, Z_1 and Z_2 are elements of $N_r \cup N_s$. We consider four cases, and in each case, derive a contradiction for G being LR(1).

CASE 1. $Z_1 \in N_r$ *and* $Z_2 \in N_r$

In this case we must have the following productions in P':

$[\text{I,C}] \rightarrow Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{l-1} Y_{l-1} X_l K$,

$[J,D] \rightarrow M_0 N_1 M_1 N_2 M_2 \cdots N_{n-1} M_{n-1} N_n Z_1$,

$[I',C'] \rightarrow Y'_0 X'_1 Y'_1 X'_2 Y'_2 \cdots X'_{l'-1} Y'_{l'-1} X'_{l'} K'$,

$[J',D'] \rightarrow M'_0 N'_1 M'_1 N'_2 M'_2 \cdots N'_{n'-1} M'_{n'-1} N'_{n'} Z_2$,

such that $l > 1$, $l' > 1$, $n \geq 0$, $n' \geq 0$, and there exists i, j, $0 < i < l$, $0 < j < l'$, where

$a \in FIRST_1(X_{i+1} Y_{i+1} X_{i+2} Y_{i+2} \cdots X_l)$,

$a \in FIRST_1(X'_{j+1} Y'_{j+1} X'_{j+2} Y'_{j+2} \cdots X'_{l'})$,

$[J,D] \in R_\varepsilon(X_i) \cup \{X_i\}$, and

$[J',D'] \in R_\varepsilon(X'_j) \cup \{X'_j\}$.

Let $Z_1 = [k_1, K_1]$ and $Z_2 = [k_2, K_2]$. Then, there must exist items $[\text{D} \rightarrow D_1 \cdots D_n . \mid a]$ in state k_1, and $[D' \rightarrow D'_1 \cdots D'_{n'} . \mid a]$ in state k_2 of **C**. We first show that $k_1 = k_2$. Three subcases are considered.

SUBCASE 1.1 $B \in N''$

Let B=[s,B'], for some $B' \in$N and $s \in$**C**. Then, we must have that $N_n = N'_{n'} = D_n = D'_{n'} = B'$, and both items

$[\text{D} \rightarrow D_1 \cdots D_{n-1} . B' \mid a]$

$[D' \rightarrow D'_1 \cdots D'_{n'-1} . B' \mid a]$

belong to the state s in **C**. Hence, $k_1 = k_2 = \text{GOTO}(s, B')$.

SUBCASE 1.2 $B \in N_S$

For this case, we must have that $n = n' = 0$ and $M_0 = M'_0 = B$. (Recall that the members of N_S can appear to the right of the members of N_T only in productions of the form $F \rightarrow IK$.) Thus, both items $[D \rightarrow . \mid \alpha]$ and $[D' \rightarrow . \mid \alpha]$ belong to state $k_1 = k_2 = B$.

SUBCASE 1.3 $B \in \Sigma$

In this case $N'_n = N'_{n'} = D_n = D'_{n'} = B$ and we must examine A. Obviously, $A = M_{n-1} = M'_{n'-1}$ is a symbol in N_S and we have that the items

$$[D \rightarrow D_1 \cdots D_{n-1} . B \mid \alpha]$$

$$[D' \rightarrow D'_1 \cdots D'_{n'-1} . B \mid \alpha]$$

belong to state A, and therefore, $k_1 = k_2 = GOTO(A,B)$.

In all of the above subcases, we have shown that $k_1 = k_2$. Thus, the items $[D \rightarrow D_1 \cdots D_n . \mid \alpha]$ and $[D' \rightarrow D'_1 \cdots D'_{n'} . \mid \alpha]$ imply a reduce/reduce conflict in state $k_1 = k_2$ in **C**. Again, this is contradictory to the assumption that G is LR(1).

CASE 2. $Z_1 \in N_S$ *and* $Z_2 \in N_S$

Since the members of N_T appear rightmost in any right-part, we must have that $B \notin N_T$. Additionally, B cannot be a symbol in N_S. This holds because if $(AB,\alpha) \in \rho(Z)$, $Z \in N_S$, then there must exist a production of the form $X \rightarrow \alpha B D \beta$ in P' such that $Z \in L(D) \cup \{D\}$; this implies that $B \notin N_S$. Therefore, we consider two subcases:

SUBCASE 2.1 $B \in N''$

If B is a symbol in N'', it must be of the form $[J,B']$ for some $J \in$ **C**, $B' \in$ N and there must exist a production

$$[\text{I,C}] \rightarrow Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{l-1} Y_{l-1} X_l K, \quad l > 0,$$

in P' such that for some i, $1 \leq i < l$, we have that $X_i = [J,B']$=B, and either $Y_i = Z_1$, or $Y_i = \varepsilon$ and $X_{i+1} = [Z_1, \text{D}]$ for some $D \in N$. Hence, Z_1=GOTO(J,B'). Similarly, we must have that Z_2=GOTO(J,B'). Therefore, Z_1 and Z_2 are not distinct and $Z_1 = Z_2$.

SUBCASE 2.2 $B \in \Sigma$

If B belongs to Σ, then we have that $A \in N_s$ and there must exist a production

$$[I,C] \rightarrow Y_0 X_1 Y_1 X_2 Y_2 \cdots X_{l-1} Y_{l-1} X_l K, \quad l > 0,$$

in P' such that for some i, $1 \leq i < l$ we have that $X_i = B$, $Y_{i-1} = A$ and either $Y_i = Z_1$ or $Z_1 \in$ **L**(X_{i+1}). This implies that Z_1=GOTO(A,B). Similarly, we have that Z_2==GOTO(A,B). Therefore, Z_1 and Z_2 are not distinct and we have a contradiction.

CASE 3. $Z_1 \in N_r$ *and* $Z_2 \in N_s$

If (AB,α)$\in \rho_{2,1}(Z_1)$, $Z_1 \in N_r$, then we must have that B$\in N'' \cup N_s \cup \Sigma$. On the other hand, if (AB,α)$\in \rho_{2,1}(Z_2)$, $Z_2 \in N_s$, then B$\in N'' \cup \Sigma$. But (AB,α) belongs to $\rho_{2,1}(Z_1) \cap \rho_{2,1}(Z_2)$, where $Z_1 \in N_r$ and $Z_2 \in N_s$; therefore, B is a symbol in $N'' \cup \Sigma$. Let $Z_1 = [k,K]$. As shown in case 1, there must exist an item [D$\rightarrow D_1 D_2 \cdots D_n$. | α], $n \geq 0$, in state k of **C**, where

$$k = \begin{cases} GOTO(J,B') & B = [J,B'] \in N''; \\ GOTO(A,B) & B \in \Sigma \end{cases}$$

Now consider (AB,α)$\in\rho_{2,1}(Z_2)$, $Z_2\in N_s$. It follows from the proof for case 2 that $Z_2=k$ and there exists an item

$$[C\rightarrow C_1 C_2 \cdots C_{i-1}.C_i \cdots C_l \mid u],\ l>0,$$

in state $Z_2=k$ in **C**, where $\alpha\in FIRST_1(C_i \cdots C_l u)$. Now if C_i does not derive ε, $\alpha\in FIRST_1(C_i \cdots C_l u)$ implies $\alpha\in EFF_1(C_i \cdots C_l u)$ and we have a shift/reduce conflict in state $Z_2=k$. On the other hand, if $\alpha\in \mathbf{L}_\varepsilon(C_i)$, then there exists F$\in$**L**($C_i$) such that the item $[F'\rightarrow .\mid \alpha]$ belongs to the state $Z_2=k$. This time we have a reduce/reduce conflict in the state $Z_2=k$. But G is LR(1) and there cannot have any conflict in state Z_2. Hence, $\rho_{2,1}(Z_1)\cap\rho_{2,1}(Z_2)=\phi$.

CASE 4. $Z_1\in N_s$ *and* $Z_2\in N_r$

The proof for this case is identical to that given for case 3.

4.2 Conversion of LR(1) Grammars to Equivalent EWP Grammars

In this section we present an algorithm for converting arbitrary LR(1) grammars into equivalent EWP grammars.

ALGORITHM 4.5 Conversion of an arbitrary LR(1) grammar to an equivalent EWP grammar.

INPUT: G=(N,Σ,P,S), an arbitrary LR(1) grammar.

OUTPUT: $G'=(N',\Sigma,P',S')$, an equivalent EWP grammar.

METHOD:

(1) Apply Algorithm 4.1 to G, giving $G_1=(N_1,\Sigma,P_1,S_1)$.

(2) $N_2=N_1$

$P_2 = \{Z\rightarrow\varepsilon \mid Z\in N_\varepsilon(G_1)\} \cup \{A\rightarrow Y_0K \mid A\rightarrow Y_0K\in P_1,\ Y_0,\ K\in N_\varepsilon(G_1)\}$

For each production $A \rightarrow Y_0X_1Y_1X_2Y_2 \cdots X_{n-1}Y_{n-1}X_nK$, $n>0$, do

(2.1) For each i, $1 \leq i \leq n$, define

$$X'_i = \begin{cases} X_i & X_i \notin \Sigma \\ [Y_{i-1}, X_i] & X_i \in \Sigma \end{cases}$$

(2.2) Add production $A \rightarrow X'_1X'_2 \cdots X'_nK$ to P_2

(2.3) For each i, $1 \leq i \leq n$, such that $X'_i = [I, a]$, for some $I \in N_s$ and $a \in \Sigma$ ($X_i \in \Sigma$) do

(2.3.1) Add $[I, a]$ and I^a to N_2,

(2.3.2) Add production $[I, a] \rightarrow I^a a$, and $I^a \rightarrow \varepsilon$ to P_2.

(3) Construct G' by reducing $G_2 = (N_2, \Sigma, P_2, S_1)$.

Before showing that G' is an EWP grammar and L(G)=L(G'), we observe that N' may be written as $N'' \cup N_\Sigma \cup N_l \cup N_s$, where N'', N_Σ, N_l and N_s are disjoint sets defined below:

$$N'' = \{[I, A] \mid I \in \mathbf{C}, A \in N, [I, A] \rightarrow \alpha \in P_1, \alpha \neq \varepsilon\}$$

$$N_\Sigma = \{[I, a] \mid a \in \Sigma, I \in N_s, A \rightarrow \alpha I a \beta \in P_1\}$$

$$N_l = \{I^a \mid a \in \Sigma, I \in N_s, [I, a] \in N_\Sigma\}$$

$$N_r = \{[k, Z] \mid k \in \mathbf{C}, Z \subseteq \Sigma \cup \{\varepsilon\}\}$$

$$N_s = \{I \mid I \in \mathbf{C}\}$$

Furthermore, if $A \rightarrow \alpha$, $\alpha \neq \varepsilon$ is a production in P' then exactly one of the following holds:

- α does not contain any symbol in $N_\Sigma \cup \Sigma$ and $A \rightarrow \alpha$ is a production in P_1.

- $\alpha=\alpha_1[I,a]\alpha_2$, $[I,a]\in N_\Sigma$, and $A\rightarrow\alpha'_1 I a\,\alpha'_2$, $\alpha'_1\alpha'_2\in V^*_{G_1}$, is a production in P_1.
- $A=[I,a]$, $\alpha=I^a a$, and $A\rightarrow\alpha$ is not a production in P_1.

LEMMA 4.6 In Algorithm 4.5, $L(G)=L(G')$.

Proof. It is clear from the algorithm that a production $A\rightarrow Y_0X_1Y_1X_2Y_2\cdots X_{m-1}Y_{m-1}X_mK$, $m\geq 0$, belongs to P_1 if and only if either $m=0$, and $A\rightarrow Y_0K\in P'$, or $m>0$, and there exists a production $A\rightarrow X'_1X'_2\cdots X'_{m-1}X'_mK$, in P', where for all i, $1\leq i\leq m$, either $X_i=X'_i\notin\Sigma$, or $X_i\in\Sigma$, and $X'_i\in\Sigma$ derives exactly X_i. Considering that for all i, $0\leq i\leq m$, Y_i derives exactly the empty string, we conclude that $L(G_1)=L(G')$. But we have already established in Lemma 4.3 that $L(G)=L(G_1)$. Therefore, $L(G)=L(G')$.

LEMMA 4.7 In Algorithm 4.5, G' is an EWP grammar.

Proof. We show that G' satisfies all of the conditions of the Definition 3.9.

1. G' is Almost Uniquely Invertible.

Let $A\rightarrow\alpha$, and $B\rightarrow\alpha$, $\alpha\neq\varepsilon$, be productions in P'. We show that A=B; that is, $A\rightarrow\alpha$ and $B\rightarrow\alpha$ are not distinct. Since $\alpha\neq\varepsilon$, $A\rightarrow\alpha$ and $B\rightarrow\alpha$ must have been added to P' in either step (2.2), or step (2.3.2) of Algorithm 2. Clearly, if both $A\rightarrow\alpha$, and $B\rightarrow\alpha$ are added to P' in step (2.3.2) of the algorithm, then we must have that A=B. This follows from the fact that all of the productions added in this step are of the form $[I,a]\rightarrow I^a a$, $I\in N_s$, $a\in\Sigma$ and if $\alpha=Y^a a$, we must have that $A=[I,a]=B$. Also productions added in the step (2.2) have some nullable symbol as the rightmost symbol in their right-parts, while the productions added in step (2.3.2) have a terminal symbol in the extreme right of their right-parts. There-

fore, both A→α and B→α should have been added to P_2 in step (2.2). Let $\alpha = X'_1 X'_2 \cdots X'_{m-1} X'_m K$, for some m, $m>0$, then there must exist productions

$$A \rightarrow Y_0 A_1 Y_1 A_2 Y_2 \cdots A_{m-1} Y_{m-1} A_m K, \text{ and}$$

$$B \rightarrow Z_0 B_1 Z_1 B_2 Z_2 \cdots B_{m-1} Z_{m-1} B_m K,$$

in P_1 that give rise to the productions A→α and B→α, respectively. Furthermore, we must have that for all i, $1 \le i \le m$, X'_i is a symbol in $(N'' \cup N_\Sigma)$. If $X'_i \in N''$, then clearly, $A_i = B_i = X'_i$ and $Y_{i-1} = Z_{i-1} = \varepsilon$. On the other hand, if $X'_i \in N_\Sigma$, then it must be a symbol of the form $[I_i, \alpha_i]$, and we have that $A_i = B_i = \alpha_i$, and $Y_{i-1} = Z_{i-1} = I_{i-1}$. Hence, for all i, $1 \le i \le m$, $A_i = B_i$ and for all i, $0 \le i \le m$, $Y_i = Z_i$. That is, the above productions have the same non-empty right-part. But G_1 is a (2,1)EWP and therefore it is Almost Uniquely Invertible. Thus A=B, and A→α and B→α are not distinct productions and G' is AUI.

2. G' is cycle-free.

Suppose for the sake of contradiction that G' has a cycle, A==>$^+$A. Clearly, if X→δ, is a production which is used in some step of the derivation A==>$^+$A, then δ does not contain a symbol in $N_\Sigma \cup \Sigma$; this implies that X→δ is also a production in P_1. It then follows that G_1 has a cycle. This contradicts the fact the G_1 is cycle-free. Hence, G' is cycle-free.

3. (1,1) EP relations are pair-wise disjoint for G'.

Suppose C $\Leftrightarrow$ D, D$\in\Sigma$. Members of N_l only appear in productions of the form A→I^αa, and therefore, we must have that C$\notin N_l$. But, if C$\stackrel{\pm}{=}D$, D$\in\Sigma$, then we must have that $C \in N_l$. Hence, ($\stackrel{\pm}{=} \cap \Leftrightarrow$)=$\phi$ for G'. Now consider a relation of the form C<$\Leftrightarrow$ D. Obviously, for all B$\in N'$, L(B)$\cap\Sigma$ is empty, and therefore we must have that D$\notin\Sigma$. Hence, ($\Leftrightarrow \cap <\Leftrightarrow$ =)ϕ for G', and the proof is complete.

4. If $A\rightarrow\alpha X\beta$, *and* $B\rightarrow\beta$ *are productions in* P', then $(X,B)\notin(\stackrel{\pm}{=}\cup\lessdot)$.

We consider two cases:

CASE 1. $\beta=\varepsilon$

Obviously, if $A\rightarrow\alpha X$ is a production in P', then X is a symbol in $N_T\cup\Sigma$. The symbols in $N_T\cup\Sigma$, however, are not related to any symbol in $V_{G'}$ by relations $\stackrel{\pm}{=}$, or $\lessdot$. Thus, for all B such that $B\rightarrow\gamma$, $\gamma\in V_{G'}^{*}$, is a production in P', $(X,B)\notin(\stackrel{\pm}{=}\cup\lessdot)$. The special case of this argument when $\gamma=\varepsilon$, establishes the desired result.

CASE 2. $\beta\neq\varepsilon$

In this case, we must have that $B\rightarrow\beta$ is not a production of the form $X\rightarrow I^{\alpha}\alpha$. Furthermore, there must exist two productions $A\rightarrow\alpha'X'\beta'$, and $B\rightarrow\beta'$ in P_1 which give rise to $A\rightarrow\alpha X\beta$ and $B\rightarrow\beta$, respectively. But, we have shown in Lemma 4.4 that if $A\rightarrow\alpha'X'\beta'$ is a production in P_1, then for no B is it true that $B\rightarrow\beta'$ is also a production in P_1. Thus, if $A\rightarrow\alpha X\beta$, $\beta\neq\varepsilon$ is a production in P', then $B\rightarrow\beta$ is not a production in P'. Hence, this case is not possible for P'.

5. For all $Z_1, Z_2\in N_{\varepsilon}(G')$, $\rho(Z_1)\cap\rho(Z_2)=\phi$.

We first observe that $N_{\varepsilon}(G')=N_I\cup N_T\cup N_S$, and if Z is a symbol in $N_T\cup N_S$, then it is also a symbol in $N_{\varepsilon}(G_1)$. Additionally, if Z_1 and Z_2, $Z_1\neq Z_2$, belong to $N_{\varepsilon}(G_1)$, and (X,α) is a pair in $\rho(Z_1)\cap\rho(Z_2)$ in G_1, then X must be a terminal symbol. That is, in G_1, two symbols of the left context are needed to resolve the conflicts between two ε-rules, only when the first symbol of the left context is a terminal symbol (see the proof of the Lemma 4.4).

Suppose for the sake of contradiction that Z_1 and Z_2, $Z_1 \neq Z_2$ are symbols in $N_\varepsilon(G')$ and there exists a pair (E,α) in $\rho(Z_1) \cap \rho(Z_2)$ in G'. First, we observe from the form of the productions in P' that E is not a symbol in $N_T \cup N_l \cup \Sigma$. Hence, E $\in (N'' \cup N_S \cup N_\Sigma)$. We consider six different cases:

CASE 1. $Z_1 \in N_T$, *and* $Z_2 \in N_T$.

If E is a symbol in N_S, then for some A and B in N'', P' must contain productions A$\rightarrow EZ_1$, and B$\rightarrow EZ_2$. Productions of this form are also productions in P_1, and therefore, (E,α)$\in \rho(Z_1) \cap \rho(Z_2)$ in G_1. But E is not a terminal symbol and we must have that $\rho_{2,1}(Z_1) \cap \rho_{2,1}(Z_2) \neq \phi$ in G_1. Hence, we have a contradiction for G_1 being (2,1)EWP.

Now suppose E is a symbol in $N_\Sigma \cup N''$. For this case, we must have productions

$$A \rightarrow A_1 A_2 \cdots A_m Z_1, \quad m \geq 1,$$

$$B \rightarrow B_1 B_2 \cdots B_n Z_2, \quad n \geq 1,$$

in P' such that $A_m = B_n =$E. Additionally, P_1 must contain productions

$$A \rightarrow Y_0 A'_1 Y_1 A'_2 Y_2 \cdots Y_{m-1} A'_m Z_1, \text{ and}$$

$$B \rightarrow Y'_0 B'_1 Y'_1 B'_2 Y'_2 \cdots Y'_{n-1} B'_n Z_2$$

which are used by Algorithm 4.5 to construct the productions

$$A \rightarrow A_1 A_2 \cdots A_m Z_1, \text{ and}$$

$$B \rightarrow B_1 B_2 \cdots B_n Z_2,$$

respectively. If E is a symbol in N'', then we must have that $A'_m = B'_n =$E. This implies that (E,α)$\in \rho(Z_1) \cup \rho(Z_2)$ in G_1. Again, E is not a terminal symbol and we have a contradiction for G_1 being (2,1)EWP. On the other hand, if E is not a symbol in N'', it should be a symbol of the form [l,b] in N_Σ. This implies that $A'_m = B'_n = b$, l$= Y_{m-1} = Y'_{n-1}$ and (lb,α)$\in \rho_{2,1}(Z_1) \cap \rho_{2,1}(Z_2)$ in G_1. Again, we

$A'_m = B'_n = b$, $l = Y_{m-1} = Y'_{n-1}$ and $(lb, a) \in \rho_{2,1}(Z_1) \cap \rho_{2,1}(Z_2)$ in G_1. Again, we have a contradiction for G_1 being (2,1)EWP.

CASE 2. $Z_1 \in N_s$, *and* $Z_2 \in N_s$.

For this case we must have productions

$A \rightarrow A_1 A_2 \cdots A_m J$, $m > 1$,

$B \rightarrow B_1 B_2 \cdots B_n J'$, $n > 1$,

$C \rightarrow Z_1[k, K]$, and

$D \rightarrow Z_2[k', K']$

in P' such that for some i, j, $1 < i \leq m$, $1 < j \leq n$, we have that $A_{i-1} = B_{j-1} = E$, $C \in L(A_i) \cup \{A_i\}$, and $D \in L(B_j) \cup \{B_j\}$. Additionally, P_1 must contain productions

$A \rightarrow Y_0 A'_1 Y_1 A'_2 Y_2 \cdots A'_{m-1} Y_{m-1} A'_m J$,

$B \rightarrow Y'_0 B'_1 Y'_1 B'_2 Y'_2 \cdots B'_{n-1} Y'_{n-1} B'_n J'$,

$C \rightarrow Z_1[k, K]$, and

$D \rightarrow Z_2[k', K']$

where $C \in L(A'_i) \cup \{A'_i\}$ and $D \in L(B'_j) \cup \{B'_j\}$. Moreover, since A'_i and B'_j are symbols of N'', we must have that $Y_{i-1} = Y'_{j-1} = \varepsilon$. Now if E is a symbol in N'', then it should hold that $A'_{i-1} = B'_{j-1} = E$, and $(E, a) \in \rho(Z_1) \cap \rho(Z_2)$ in G_1. But, E is not a terminal symbol and similar to case 1, we have a contradictions for G_1 being (2,1)EWP.

If E is not a symbol in N'', however, it must be a symbol of the form $[l, b]$ in N_Σ. This implies that $A'_{i-1} = B'_{j-1} = b$, $l = Y_{i-2} = Y'_{j-2}$ and we must have that $(lb, a) \in \rho_{2,1}(Z_1) \cap \rho_{2,1}(Z_2)$ in G_1. Again, we have a contradiction for G_1 being (2,1)EWP.

CASE 3. $Z_1 \in N_l$, *and* $Z_2 \in N_l$.

Members of N_l appear only in productions of the form $[I,b] \rightarrow I^b b$. Therefore, Z_1 and Z_2 must be of the forms E_1^a, and E_2^a, respectively. Furthermore, P' must contain productions

$A \rightarrow A_1 A_2 \cdots A_m K, \quad m>1,$

$B \rightarrow B_1 B_2 \cdots B_n K', \quad n>1,$

$[E_1,a] \rightarrow E_1^a a$, and

$[E_2,a] \rightarrow E_2^a a$

such that for some i and j, $1<i \leq m$, $1<j \leq n$, we have that $A_{i-1}=B_{j-1}=E$, $[E_1,a] \in L(A_i) \cup \{A_i\}$, and $[E_2,a] \in L(B_j) \cup \{B_j\}$. Therefore, in P_1 we must have productions

$A \rightarrow Y_0 A'_1 Y_1 A'_2 Y_2 \cdots A'_{m-1} Y_{m-1} A'_m K,$

$B \rightarrow Y'_0 B'_1 Y'_1 B'_2 Y'_2 \cdots B'_{n-1} Y'_{n-1} B'_n K',$

such that $a \in L_\varepsilon(A'_i) \cup \{A'_i\}$, $a \in L_\varepsilon(B'_j) \cup \{B'_j\}$, and exactly one of the following holds

- $Y'_{j-1}=Y_{i-1}=\varepsilon$, and $E=[I,E']=A'_{i-1}=B'_{j-1}$ is a symbol in N''.
- $Y'_{j-1}=Y_{i-1} \in N_s$, $A'_{i-1}=B'_{j-1}=b$, $b \in \Sigma$, and $E=[I,b] \in N_\Sigma$, where $I=Y_{i-2}=Y'_{j-2}$ and $A'_{i-1}=B'_{j-1}=b$.

If the first condition above holds, then we must have that $E_1=E_2=\text{GOTO}(I,E')$. On the other hand, if the second condition holds, we must have that $E_1=E_2=\text{GOTO}(I,b)$. Therefore, $E_1^a=E_2^a$ and we have a contradiction to the assumption that $Z_1=E_1^a \neq Z_2=E_2^a$.

CASE 4. $Z_1 \in N_r$, *and* $Z_2 \in N_s$.

For this case, there must exist productions

$A \rightarrow A_1 A_2 \cdots A_m Z_1, \quad m \geq 1,$

$B \rightarrow B_1 B_2 \cdots B_n K, \quad n > 1,$ and

$C \rightarrow Z_2[k, K]$

in P', such that for some i, $1 < i \leq n$, $C \in \mathbf{L}(B_i) \cup \{B_i\}$, and $A_m = B_{i-1} = E$. Moreover, P_1 must contain productions

$A \rightarrow Y_0 A'_1 Y_1 A'_2 A Y_2 \cdots A'_{m-1} Y_{m-1} A'_m Z_1,$

$B \rightarrow Y'_0 B'_1 Y'_1 B'_2 Y'_2 \cdots B'_{n-1} Y'_{n-1} B'_n K,$

$C \rightarrow Z_2[k, K]$

where $C \in \mathbf{L}(B'_i) \cup \{B'_i\}$, and either E is a symbol in N'', and $A'_m = B'_{i-1} = E$; or E is a symbol of the form $[I, b]$ in Σ, $A'_m = B'_{i-1} = b$, and $I = Y_{m-1} = Y'_{i-2}$. In either case, it follows form the construction that G has two items

$[B' \rightarrow B_1 \text{''} B_2 \text{''} \cdots B_n \text{''}. \mid a],$

$[C \rightarrow . \mid a]$

in state J of **C**, where J is defined below:

$$J = \begin{cases} GOTO(I, E') & E = [I, E'] \in N''; \\ GOTO(I, b) & E = [J, b] \in N_{\Sigma}; \end{cases}$$

The existence of these items in the same state in **C** implies that G is not LR(1). But G is assumed to be LR(1) and we have a contradiction.

CASE 5. $Z_1 \in N_s$, *and* $Z_2 \in N_t$.

In this case, P' must contain productions

$A \rightarrow A_1 A_2 \cdots A_m J, \quad m \geq 1,$

$B \rightarrow B_1 B_2 \cdots B_n J', \quad n > 1,$

$C \rightarrow Z_1[k, K],$ and

$D \rightarrow Z_2 a$

such that for some i and j, $1<i\leq m$, $1<j\leq n$, $A_{i-1}=B_{j-1}=E$, D$\in$**L**$(A_i)\cup\{A_i\}$, and C$\in$**L**$(B_j)\cup\{B_j\}$. Hence, P_1 must contain productions

$$A\rightarrow Y_0A'_1Y_1A'_2AY_2\cdots A'_{m-1}Y_{m-1}A'_mJ,$$

$$B\rightarrow Y'_0B'_1Y'_1B'_2Y'_2\cdots B'_{n-1}Y'_{n-1}B'_nJ',$$

$$C\rightarrow Z_1[k,K]$$

such that C$\in$**L**$(B'_j)\cup\{B'_j\}$ and $\alpha\in$**L**$_\varepsilon(A'_i)\cup\{A_i\}$. Here, using an argument similar to the one given for the previous case, we can show that some state in **C** contains a pair of items

$$[A'\rightarrow A''_1A''_2\cdots A''_{i-1}.A''_i\cdots A''_n\,|\,v]$$

$$[C'\rightarrow .\,|\,\alpha]$$

such that $\alpha\in EFF_1(A''_i)$. Obviously, this is contradictory to the assumption that G is LR(1).

CASE 6. $Z_1\in N_r$, *and* $Z_2\in N_l$.

Combining the arguments for cases (4) and (5) , we conclude that some state in **C** contains items

$$[A'\rightarrow A''_1A''_2\cdots A''_{i-1}.A''_i\cdots A''_n\,|\,v],\ \text{and}$$

$$[B'\rightarrow B''_1B''_2\cdots B''_n.\,|\,\alpha]$$

such that $\alpha\in EFF_1(A''_i)$. Again, we have a contradiction for G being LR(1).

We conclude this section by a stating a corollary to Lemma 4.7.

COROLLARY Every deterministic context-free language is defined by some EWP grammar.

Proof. This is a direct result of the fact that the class of LR(1) grammars define exactly the class of deterministic context-free languages, and that Algorithm 4.5 converts arbitrary LR(1) grammars into equivalent EWP grammars.

4.3 Viable Prefix EWP Grammars and Parsers

The purpose of this section is the development of EWP parsers with the viable prefix property. We define the class of Viable Prefix EWP grammars and show that the EWP parser for every Viable Prefix EWP grammar has the viable prefix property. Additionally, we establish that every deterministic context-free language is defined by some Viable Prefix EWP grammar.

First, the concept of valid nonterminals used to define the class of Viable Prefix EWP grammars is introduced.

DEFINITION 4.8 Let G=(N,Σ,P,S) be a CFG. We say that X∈N is valid for $\gamma \subset V_G^*$, if there exists a derivation $S ==>_{rm}^{*} \gamma X \omega$, $\omega \subset \Sigma^*$ in G.

We now define the class of Viable Prefix EWP grammars.

DEFINITION 4.9 Let G=(N,Σ,P,S) be an EWP grammar. Define the set, $\Pi(\gamma)$ as follows:

$\Pi(\gamma)$={(A→$\alpha\beta$, α) | there is a derivation
$S ==>_{rm}^{*} \delta A \omega ==> \delta\alpha\beta\omega = \gamma\beta\omega$ in G}.

G is said to be a *Viable Prefix* EWP grammar if the following holds:

(1) If A→$\alpha a \beta$, $a \in \Sigma$, is a production in P, then $\beta = \varepsilon$;

(2) Let X∈N be valid for $\gamma \in V_G^*$, and A→$\alpha X \beta$∈P. Then, the pair ($B \rightarrow \delta X \beta$, δ) belongs to $\Pi(\gamma)$, where $B \rightarrow \delta X \beta$∈P.

In our next theorem we establish that EWP parsers for Viable Prefix EWP grammars have the viable prefix property.

THEOREM 4.10 The EWP parser for any Viable Prefix EWP grammar has the viable prefix property.

Proof. Let G=(N,Σ,P,S) be a Viable Prefix EWP grammar and **P** be the EWP parser for G. Assume that ω is a string in Σ^* and $(\$, \omega\$) \mid --\mathbf{P}^n (\$\gamma , \omega'\$)$. We prove by induction on n that γ is a viable prefix of G.

BASIS: $n=1$. For $n=1$, we have that $(\$, \omega\$) \mid --_{\mathbf{P}} (\$\gamma , \omega'\$)$ and either $\gamma=a=PREF_1(\omega)\in\Sigma$, or $\gamma=I\in N_\varepsilon(G)$ and $(\$, PREF_1(\omega))\in\rho(I)$. In either case, we must have that $(\$,\gamma)\in\lessdot$ and $\gamma\in\mathbf{L}(S)$. Hence, there must exist a derivation $S ==>_{rm}^{*} Ay ==>\gamma\delta y$ in G and therefore, γ is a viable prefix of G.

INDUCTIVE STEP: Assume that the theorem is true for all $m<n$, $n>1$ and consider $(\$, \omega\$) \mid --\mathbf{P}^{n-1} (\$\gamma' , y\$) \mid --(\$\gamma , \omega'\$)$. Let $\gamma'=\gamma''X$, $\gamma''\in V_G^*$, and $y=ay'$, $y'\in\Sigma^*$. From the inductive hypothesis, we have that $\gamma''X$ is a viable prefix of G. Therefore, there must exist a derivation $S ==>_{rm}^{*} \vartheta Au ==>_{rm} \vartheta\alpha_1 X\alpha_2 u =\gamma''X\alpha_2 u ==>_{rm}^{*} \gamma''Xu'$ in G. Additionally, $(\$\gamma''X , ay'\$) \mid --(\$\gamma , \omega'\$)$ and we must have that $(X,a)\in(\doteq \cup \lessdot \cup \gtrdot)$. Two cases are considered:

CASE 1. $(X,a)\in(\doteq \cup \lessdot)$.

Since $(X,a)\in(\doteq \cup \lessdot)$, there must exist a production $B\rightarrow\alpha XY\beta$ in P such that $a\in\mathbf{L}(Y)\cup\{Y\}$. Hence, it follows from condition (1) of Definition 4.9 that X∈N. Thus, X is valid for γ'' and it follows from condition (2) of Definition 4.9 that there exists a pair $(C\rightarrow\alpha'XY\beta,\alpha')$ in $\Pi(\gamma'')$. This implies that there exists a derivation

$$S ==>_{rm}^{*} \vartheta Cv ==>_{rm} \vartheta\alpha'XY\beta v =\gamma''XY\beta v ==>_{rm}^{*} \gamma''XYv' ==>_{rm}^{*} \gamma''Xav''$$

in G. Clearly, this derivation implies that $\gamma''Xa=\gamma$ is a viable prefix of G and the proof for this case is complete.

CASE 2. $(X,\alpha)\in\Leftrightarrow$.

In this case, the configuration $(\$\gamma,\omega'\$)$ is derived from $(\$\gamma',y\$)=(\$\gamma''X,\alpha y'\$)$ by a reduce move. Therefore, we must have that $y=\omega'$, $\gamma''X=\delta\beta$ and $\gamma=\delta A$, where $A\rightarrow\beta\in P$. That is, $A\rightarrow\beta$ is the production used for the reduction. Two subcases are considered:

SUBCASE 2.1 $\beta=\varepsilon$.

If $\beta=\varepsilon$, then we have that $(X,\alpha)\in\rho(A)$. Hence, $(X,A)\in(\stackrel{+}{=}\cup\Leftarrow)$ and there must exist a production $B\rightarrow\alpha XY\beta'$ in P such that $A\in\mathbf{L}(Y)\cup\{Y\}$. Moreover, it follows from condition (1) of Definition 4.9 that X∈N. Considering that X is valid for γ'', we conclude that there exists a pair $(B'\rightarrow\alpha'XY\beta',\alpha')$ in $\Pi(\gamma'')$. That is, there exists a derivation

$$S==>^{*}_{rm}\delta B'u==>\delta\alpha'XY\beta'u=\gamma''XY\beta'u==>^{*}_{rm}\gamma''XYu'==>^{*}_{rm}\gamma''XA\lambda u'$$

in G. Thus, $\gamma''XA=\gamma$ is a viable prefix of G.

SUBCASE 2.2 $\beta\neq\varepsilon$.

For this case, we have that $\beta=\beta'X$, $\beta'\in V_G^{*}$. If $X\in\Sigma$, then since γ'' is a viable prefix of G, there must exist a derivation

$$S==>^{*}_{rm}\alpha Bu==>_{rm}\alpha\beta''X\lambda u=\gamma''X\lambda u$$

in G. But G is a Viable Prefix EWP grammar and the terminals appear only rightmost in any production. Therefore, $\lambda=\varepsilon$. On the other hand, if $X\in N$, then it is a valid for γ'' and it follows from condition (2) of Definition 4.9 that there is a pair $(B\rightarrow\beta''X,\beta'')$ in $\Pi(\gamma'')$. Again,

$$S==>^{*}_{rm}\alpha Bu==>_{rm}\alpha\beta''Xu=\gamma''Xu$$

is a derivation in G.

Now consider the productions $A \rightarrow \beta' X$ and $B \rightarrow \beta'' X$. Clearly, both $\beta' X$ and $\beta'' X$ are suffixes of $\gamma'' X$ and we must have that either β' or β'' is a suffix of the other. But, the EWP parser always reduces the stack by the production having the longest right part and $A \rightarrow \beta' X$ is the production selected to reduce $\gamma'' X$. Therefore, $\beta' X = \vartheta \beta'' X$ for some $\vartheta \in V_G^*$. If $\vartheta \neq \varepsilon$, then $\vartheta = \vartheta' Y$ and by applying Lemma 3.6 to the derivation

$$S ==>_{rm}^{*} \alpha B u ==> \alpha \beta'' X u = \gamma'' X u = \delta \beta' X u = \delta \vartheta \beta'' X u = \delta \vartheta' Y \beta'' X u$$

we have that $(B, Y) \in (\stackrel{\pm}{=} \cup \lessdot)$. Obviously, this is a violation of condition (4) of Definition 3.9. Hence, we must have that $\vartheta = \varepsilon$, $\alpha = \delta$ and $\beta' X = \beta'' X$. But G is AUI and $\beta' X = \beta'' X \neq \varepsilon$. Hence, A=B and it follows from the derivation

$$S ==>_{rm}^{*} \alpha B u = \delta B u = \delta A u ==>_{rm} \delta \beta' X u$$

that $\delta A = \gamma$ is a viable prefix of G.

We now prove that the grammar G' constructed by Algorithm 4.5 is a Viable Prefix EWP grammar.

THEOREM 4.11 In Algorithm 4.5, G' is a Viable Prefix EWP grammar.

Proof. Obviously, in G' terminals appear only in productions of the form $[I, a] \rightarrow I^{a} a$, $I \in \mathbf{C}$, $a \in \Sigma$ and G' satisfies condition (1) of Definition 4.9. Let $\gamma \in V_{G'}^*$ be a viable prefix of G' and $X \in N'$ be valid for γ. Let $A \rightarrow \alpha X \beta$ be a production in P'. We prove that there exists a pair $(A' \rightarrow \alpha' X \beta, \alpha')$ in $\Pi(\gamma)$. Two cases are considered:

CASE 1. $X \in N_l \cup N_r$.

Since X is valid for γ, then there is a derivation

$$S' ==>^{*}_{rm} \vartheta B \omega ==>_{rm} \vartheta \alpha' X \beta' \omega = \gamma X \beta' \omega \text{ in } G'.$$

Therefore, $(B \rightarrow \alpha' X \beta' , \alpha') \in \Pi(\gamma)$. Now consider the productions $A \rightarrow \alpha X \beta$ and $B \rightarrow \alpha' X \beta'$. As shown in theorem 4.7, each symbol of N_T appears exactly in one production and therefore, if $X \in N_T$, then A=B, $\alpha = \alpha'$ and $\beta = \beta' = \varepsilon$. On the other hand, if $X \in N_I$, then it is a symbol of the form I^a for some $I \in \mathbf{C}$ and $a \in \Sigma$. This time, we must have that A=B=$[I,a]$, $\alpha = \alpha' = \varepsilon$ and $\beta = \beta' = a$. Thus, $(A \rightarrow \alpha X \beta , \alpha) = (A \rightarrow \alpha' X \beta' , \alpha)$ is a pair in $\Pi(\gamma)$ and the proof for this case is complete.

CASE 2. $X \in N'' \cup N_s \cup N_\Sigma$.

The result for this case is established by an induction on the length of γ.

BASIS: $|\gamma| = 0$.

If $|\gamma| = 0$, then X is valid for ε and there is a derivation $S' ==>^{*}_{rm} B \omega ==>_{rm} X \beta' \omega$ in G'. Hence, $X \in \mathbf{L}(S')$ and either $X = I_0 \in N_s$ or $X = [I_0, Y] \in N'' \cup N_\Sigma$, where I_0 is the initial state in **C**. Thus, α is the empty string; otherwise, we must have that $SUFF_1(\alpha) = [J,Z]$ where GOTO$(J,Z) = I_0$, a contradiction for **C** being the canonical collection of LR(1) states for G. Thus, A=$[I_0,A'] \in \mathbf{L}(S') \cup (S')$ and there exists a derivation $S' ==>^{*}_{rm} Av ==>_{rm} X \beta v$ in G'. Hence, $(A \rightarrow X \beta , \varepsilon) \in \Pi(\varepsilon)$.

INDUCTIVE HYPOTHESIS

Assume that the theorem is true for all $\gamma \in V^{*}_{G'}$ such that $|\gamma| < n$, $n > 0$ and consider a derivation $S' ==>^{*}_{rm} \gamma X \omega$ in G', where $|\gamma| = n$. This derivation may be written as

$$S' = \delta_0 ==>_{rm} \delta_1 ==>_{rm} \delta_2 ==>_{rm} \cdots ==>_{rm} \delta_{m-1} ==>_{rm} \delta_m = \gamma X \omega,$$

where for all j, $0 \leq j \leq m$, $\delta_j = \lambda_j B_j \omega_j$, $\lambda_j \in V^{*}_{G'}$, $B_j \in N'$ and $\omega_j \in \Sigma^{*}$. Let t be the

largest integer such that γ is not a prefix of δ_t. Obviously, such a t exists since $\delta_m = \gamma X \omega$ and $|\gamma| > 0$. Let $B \rightarrow \gamma_1 C \gamma_2$, $\gamma_1 \neq \varepsilon$, be the production used to derive δ_{t+1} from δ_t. Then, $\gamma = \lambda_t \gamma_1$, $X \in \mathbf{L}(C) \cup \{C\}$ and $SUFF_1(\gamma_1)$ is valid for γ' where $\gamma' = PREF_{|\gamma|-1}(\gamma)$. Moreover, $X \notin N_T$ and we must have that $\gamma_2 \neq \varepsilon$. Therefore, $B \rightarrow \gamma_1 C \gamma_2$ is of the form

$$[J_1, B'] \rightarrow [J_1, B_1][J_2, B_2] \cdots [J_l, B_l][J, Z], \quad l > 1$$

where $C = [J_j, B_j]$, for some j, $1 < j \leq l$ and for all k, $1 \leq k \leq l$, the LR(1) item

$$[B' \rightarrow B_1 B_2 \cdots B_{k-1} . B_k \cdots B_l \mid v], \; v \in Z,$$

belongs to the state $J_k = GOTO(J_1, B_1 B_2 \cdots B_{k-1})$ in **C**.

Now consider the production A$\rightarrow \alpha X \beta$. Two subcases arise:

SUBCASE 1. $\alpha \neq \varepsilon$.

If $\alpha \neq \varepsilon$, then $X \in N'' \cup N_\Sigma$ and A$\rightarrow \alpha X \beta$ is of the form

$$[I_1, A'] \rightarrow [I_1, A_1][I_2, A_2] \cdots [I_{l'}, A_{l'}][I, Z'], \quad l' > 1$$

where $X = [I_i, A_i]$, for some i, $1 < i \leq l'$ and for all k, $1 \leq k \leq l'$, the LR(1) item

$$[A' \rightarrow A_1 A_2 \cdots A_{k-1} . A_k \cdots A_{l'} \mid u], \; u \in Z'$$

belongs to the state $I_k = GOTO(I_1, A_1 A_2 \cdots A_{k-1})$ in **C**. Additionally, $X = [I_i, A_i] \in \mathbf{L}(C) \cup \{C\}$, $C = [J_j, B_j]$ and we must have that $I_i = J_j$. Now consider the items

$$[B' \rightarrow B_1 B_2 \cdots B_{j-2} . B_{j-1} \cdots B_l \mid v] \in J_{j-1}, \text{ and}$$
$$[A' \rightarrow A_1 A_2 \cdots A_{i-2} . A_{i-1} \cdots A_{l'} \mid u] \in I_{i-1}.$$

Clearly, since GOTO(I_{i-1}, A_{i-1})=GOTO$(J_{j-1}, B_{j-1}) = I_i = J_j$, we must have that $A_{i-1} = B_{j-1}$ and $[A' \rightarrow A_1 A_2 \cdots A_{i-2} . A_{i-1} \cdots A_{l'} \mid u]$ also belongs to the state J_{j-1}. Let $[A' \rightarrow . A_1 A_2 \cdots A_{l'} \mid u] \in K_1$, where GOTO $(K_1, A_1 \cdots A_{i-2}) = J_{j-1}$. Then, it follows from the construction that there exists a production

$$[K_1,A']\rightarrow[K_1,A_1][K_2,A_2]\cdots[K_{l'},A_{l'}][K,Z'']$$

where GOTO$(K_1,A_1\cdots A_{l'})=K$, $J_{j-1}=K_{i-1}$ and for all k, $1\leq k<l'$, K_{k+1}=GOTO(K_k,A_k). But K_i=GOTO$(K_1,A_1A_2\cdots A_{i-1})=J_j=I_i$ and therefore, for all k, $i\leq k\leq l'$, $K_k=I_k$ and $[K,Z'']=[I,Z']$. That is, $[K_1,A']\rightarrow[K_1,A_1][K_2,A_2]\cdots[K_{l'},A_{l'}][K,Z'']$ is of the form $D\rightarrow\alpha'X\beta$, where $\alpha'\neq\varepsilon$ and $SUFF_1(\alpha')=SUFF_1(\gamma_1)=[J_{j-1},B_{j-1}]=[J_{j-1},A_{i-1}]$.

Let $Y=SUFF_1(\gamma_1)$. Then, Y is valid for γ', $\gamma=\gamma'Y$ and $|\gamma'|=n-1$. Therefore, from the inductive hypothesis, we have that there is a pair $(D'\rightarrow\alpha''YX\beta\ ,\ \alpha'')$ in $\Pi(\gamma')$. Hence, there is a derivation

$$S'==>^{*}_{rm}\vartheta D'\omega'==>_{rm}\vartheta\alpha''YX\beta\omega'=\gamma'YX\beta\omega'=\gamma X\beta\omega'$$

in G'. This derivation implies that the pair $(D'\rightarrow\alpha''YX\beta\ ,\ \alpha''Y)$ belongs to $\Pi(\gamma'Y)=\Pi(\gamma)$ and we have the desired result.

SUBCASE 2. $\alpha=\varepsilon$.

For this case, either X=$I_1\in N_s$ and A$\rightarrow\alpha X\beta$ is of the form $[I_1,A']\rightarrow I_1[I_1,Z']$ and $[A'\rightarrow.\,|\,u]\in I_1$, $u\in Z'$, or X=$[I_1,A_1]\in N''\cup N_\Sigma$ and A$\rightarrow\alpha X\beta$ is of the form

$$[I_1,A']\rightarrow[I_1,A_1][I_2,A_2]\cdots[I_{l'},A_{l'}][I,Z'],\quad l'>1$$

and $[A'\rightarrow.A_1A_2\cdots A_{l'}\,|\,u]\in I_1$, $u\in Z'$. Moreover, similar to the previous case, we have that I_1=GOTO$(J_{j-1},B_{j-1})=J_j$. Therefore, I_1 is not the initial state of **C** and there must exist an LR(1) item $[D'\rightarrow D_1D_2\cdots D_{i-1}.D_i\cdots D_{l''}\,|\,u]$, $l''>1$, in J_j where $1<i\leq l''$ and $A'\in$**L**$(\mathbf{D}_i)\cup\{D_i\}$. Hence, there must exist a production $D\rightarrow\alpha'E\beta'\in P'$, $\alpha'\neq\varepsilon$, A$\in$**L**$(E)\cup\{E\}$ which is constructed from the item $[D'\rightarrow.D_1D_2\cdots D_{l''}\,|\,x]\in K_1$, where GOTO$(K_1,D_1D_2\cdots D_{i-1})=J_j$. Therefore, from the previous subcase, we have that $(D''\rightarrow\alpha''E\beta',\alpha')$ is a pair in $\Pi(\gamma)$. That is, there exists a derivation $S'==>^{*}_{rm}\vartheta D''\omega'==>_{rm}\vartheta\alpha''E\beta'\omega'=\gamma E\beta'\omega'$ in G'. But A$\in$**L**$(E)\cup\{E\}$. Hence, there exists a derivation

$$S' ==>^{*}_{rm} \gamma E \beta' \omega' ==>^{*}_{rm} \gamma E' \omega'' ==>^{*}_{rm} \omega A x ==>_{rm} \gamma \alpha x$$

in G' Thus, $(A \rightarrow X\beta, \varepsilon) \in \Pi(\gamma)$.

The result established in the previous theorem is of particular importance since it shows that every deterministic context-free language has a Viable Precedence EWP grammar. This is formally stated as a corollary to Theorem 4.11.

COROLLARY Every deterministic context-free language is defined by some Viable Prefix EWP grammar.

Proof. We only observe that the class of LR(1) grammars defines exactly the class of deterministic context-free languages, and that Algorithm 4.5 converts arbitrary LR(1) grammars into equivalent Viable Prefix EWP grammars.

CHAPTER 5

PROPERTIES OF EPSILON SIMPLE PRECEDENCE LANGUAGES

Given a language, L, it is relatively easy to show that L is defined by a grammar in a particular class. One can write a grammar defining exactly L, and show that the grammar possesses all of the required properties.

On the other hand, proving that L is not in a particular class of languages poses some difficulties. It must be shown that no grammar having the required properties may define exactly L. Closure properties and iteration theorems are generally used for this purpose.

Closure properties of a class of languages, X, state whether or not X is closed under different set operations. Iteration theorems usually state necessary conditions that, if satisfied by some sentence of the language, guarantee the membership of a set (usually infinite) of sentences in the language. Iteration theorems exist for context-free [9], deterministic context-free [37], LL(k) [10], strict deterministic [25], and simple precedence languages [28].

In the remainder of this chapter, by generalizing the iteration theorem for simple precedence languages, we develop an iteration theorem for ESP_i languages. Using this theorem, the hierarchy of ESP_i languages is established. Additionally, some closure properties of ESP_i languages are studied in this chapter.

5.1 Iteration Theorem for ESP_i Languages

Before presenting the iteration theorem for ESP_i languages, some properties of derivations in ESP_i grammars are established by proving four lemmas. These lemmas generalize similar results established in [28] for simple precedence languages. Our proofs are identical to those in [28], except for the discussions relating to the use of ε-rules. Complete proofs, however, are given for the sake of completeness.

The following notations are used in this section. Let α, $\beta \in V^*$ and $\mu \in (\stackrel{\pm}{=} \cup \lessdot \cup \gtrdot)$. The relation $(SUFF_1(\alpha), PREF_1(\beta)) \in \mu$ is denoted $\alpha \mu \beta$. We say that α has only μ, if every two adjacent symbols in α are related only by μ.

In the first lemma we show that the last steps of two derivations producing right sentential forms $\alpha\beta\omega$ and $\alpha'\beta\omega'$ are identical given that the handle of both sentential forms is a substring of β and $PREF_1(\omega)=PREF_1(\omega')$, and $SUFF_1(\$\alpha) = SUFF_1(\$\alpha')$, if the handle is the empty string located in the extreme left of β.

LEMMA 5.1 Let G=(N,Σ,P,S) be an ESP grammar in which there exist derivations

(5.1.1) $\$S\$ \Longrightarrow^{*}_{rm} \$\alpha\delta\omega\$ \Longrightarrow_{rm} \$\alpha\beta\omega\$$;

(5.1.2) $\$S\$ \Longrightarrow^{+}_{rm} \$\alpha'\beta\omega'\$$;

such that

(1) $PREF_1(\omega\$)=PREF_1(\omega'\$)$;

(2) Either $\$\alpha' \lessdot \beta$ and ($\$\alpha \lessdot \beta$ or $\$\alpha \stackrel{\pm}{=} \beta$), or $SUFF_1(\$\alpha')=SUFF_1(\$\alpha)$;

(3) α' has only $\lessdot$ and $\stackrel{\pm}{=}$.

Then derivation (5.1.2) is of the form

(5.1.3) $\$S\$ ==>_{rm}^{*} \$\alpha'\delta\omega'\$ ==>_{rm} \$\alpha'\beta\omega'\$$.

Proof. Since $\delta ==>_{rm} \beta$, we must have $\delta=\gamma Ax$, and $\beta=\gamma\vartheta x$, for some ϑ, $\gamma\in V_G^*$, A∈N, and $x\in\Sigma^*$, where A→ϑ is a production in P. Therefore, derivation (5.1.1) may be rewritten as follows:

(5.1.4) $\$S\$ ==>_{rm}^{*} \$\alpha\gamma Ax\,\omega\$ ==>_{rm} \$\alpha\gamma\vartheta x\,\omega\$ = \$\alpha\beta\omega\$$.

Applying Lemma 3.6 to derivation (5.1.4), we conclude that the following hold.

(*i*) $\$\alpha\gamma$ has only $\lessdot$ and $\doteq$;

(*ii*) $\$\alpha\gamma\vartheta \rightarrowtriangle x\,\omega\$$;

(*iii*) Either $\vartheta\neq\varepsilon$, $\$\alpha\gamma \lessdot \vartheta$ and ϑ has only $\doteq$; or, $\vartheta=\varepsilon$, $\$\alpha\gamma \rightarrowtriangle x\,\omega\$$, and $(SUFF_1(\$\alpha\gamma), PREF_1(x\,\omega\$))\in\rho$(A).

Considering conditions (1)-(3) and (*i*)-(*iii*) above, the following must hold for the right sentential form $\$\alpha'\beta\omega'\$ = \$\alpha'\gamma\vartheta x\,\omega'\$$.

(*iv*) $\$\alpha'\gamma$ has only $\lessdot$ and $\doteq$;

(*v*) $\$\alpha'\gamma\vartheta \rightarrowtriangle x\,\omega'\$$;

(*vi*) Either $\vartheta\neq\varepsilon$, $\$\alpha'\gamma \lessdot \vartheta$ and ϑ has only $\doteq$; or, $\vartheta=\varepsilon$, $\$\alpha'\gamma \rightarrowtriangle x\,\omega'\$$ and $(SUFF_1(\$\alpha'\gamma), PREF_1(x\,\omega'\$))\in\rho$(B) for some B$\in N_\varepsilon(G)$.

Thus, ϑ is the handle of $\$\alpha'\beta\omega'\$ = \$\alpha'\gamma\vartheta x\,\omega'\$$, and the following derivation exists in G.

(5.1.5) $\$S\$ ==>_{rm}^{*} \$\alpha'\gamma Bx\,\omega'\$ ==>_{rm} \$\alpha'\gamma\vartheta x\,\omega'\$$,

where B→ϑ∈P. Here, we consider two cases:

CASE 1 $\vartheta\neq\varepsilon$.

A→ϑ and B→ϑ are productions in P, and since G is AUI, it immediately follows that A=B.

CASE 2 $\vartheta=\varepsilon$.

For this case, we have that B$\in N_\varepsilon(G)$, and

$$(SUFF_1(\$\alpha'\gamma),PREF_1(x\,\omega'\$))\subset\rho(B).$$

If $\gamma\neq\varepsilon$, then clearly, $SUFF_1(\$\alpha'\gamma)=SUFF_1(\$\alpha\gamma)$. On the other hand, if $\gamma=\varepsilon$, then from conclusion (vi), we have that $\$\alpha'\gamma\vartheta=\$\alpha'\Rightarrow x\,\omega'\$$, and therefore, from condition (2) of the lemma we have that $SUFF_1(\$\alpha')=SUFF_1(\$\alpha)$. Hence,

$$(SUFF_1(\$\alpha'\gamma),PREF_1(x\,\omega'\$)) = (SUFF_1(\$\alpha\gamma),PREF_1(x\,\omega\$)),$$

and $\rho(A)\cap\rho(B)\neq\phi$. But G is ESP; therefore, A=B.

Thus, in either case derivation (5.1.5) is of the form

(5.1.6) $\$S\$ \Longrightarrow^*_{rm} \$\alpha'\gamma Bx\,\omega'\$=\$\alpha'\gamma Ax\,\omega'\$=\$\alpha'\delta\omega'\$\Longrightarrow_{rm}$
$\$\alpha'\gamma\vartheta x\,\omega'=\$\alpha'\beta\omega'\$.$

That is, derivation (5.1.2) is of the form

$$\$S\$ \Longrightarrow^*_{rm} \$\alpha'\delta\omega'\$\Longrightarrow^*_{rm} \$\alpha'\beta\omega'\$,$$

and the proof is complete.

Our next lemma generalizes Lemma 5.1 by establishing that the substring β of two right sentential forms $\alpha\beta\omega$ and $\alpha'\beta\omega'$ is derived identically from some substring δ, provided that $PREF'_1(\omega)=PREF'_1(\omega')$, the handle of both sentential forms is a substring of β, and $SUFF_1(\$\alpha)=SUFF_1(\$\alpha')$, if β is the empty string.

LEMMA 5.2 Let G=(N,Σ,P,S) be an ESP grammar in which there exists derivations

(5.2.1) $\$S\$ \Longrightarrow^*_{rm} \$\alpha\delta\omega\$\Longrightarrow^n_{rm} \$\alpha\beta\omega\$$;

(5.2.2) $\$S\$ \Longrightarrow^*_{rm} \$\alpha'\beta\omega'\$$;

such that

(1) $PREF_1(\omega\$)=PREF_1(\omega'\$)$;

(2) Either $\$\alpha' \lessdot \beta$ and ($\$\alpha \lessdot \beta$ or $\$\alpha \stackrel{\pm}{=} \beta$), or $SUFF_1(\$\alpha)=SUFF_1(\$\alpha')$;

(3) α' has only $\lessdot$ and $\stackrel{\pm}{=}$.

Then derivation (5.2.2) is of the form

(5.2.3) $\$S\$ ==>_{rm}^{*} \$\alpha'\delta\omega'\$ ==>_{rm}^{n} \$\alpha'\beta\omega'\$$.

Proof. We prove this result by induction on n.

BASIS: n=0. Then $\delta=\beta$, and the result holds.

INDUCTIVE STEP: Assume that the result holds for all k, $k<n$, $n>0$, and consider the following derivation:

(5.2.4) $\$S\$ ==>_{rm}^{*} \$\alpha\delta\omega\$ ==>_{rm}^{n-1} \$\alpha\delta_{n-1}\omega\$ ==>_{rm} \$\alpha\beta\omega\$$.

Obviously, derivations (5.2.2) and (5.2.4) written as

(5.2.5) $\$S\$ ==>_{rm}^{*} \$\alpha\delta_{n-1}\omega\$ ==>_{rm} \$\alpha\beta\omega\$$.

satisfy the conditions of Lemma 5.1. Hence, we apply Lemma 5.1 to these derivations and conclude that derivation (5.2.2) is of the form

(5.2.6) $\$S\$ ==>_{rm}^{*} \$\alpha'\delta_{n-1}\omega'\$ ==>_{rm} \$\alpha'\beta\omega'\$$.

Now consider derivations:

(5.2.7) $\$S\$ ==>_{rm}^{*} \$\alpha\delta\omega\$ ==>_{rm}^{n-1} \$\alpha\delta_{n-1}\omega\$$; and

(5.2.8) $\$S\$ ==>_{rm}^{*} \$\alpha'\delta_{n-1}\omega'\$$

(these are drawn from derivations (5.2.4) and (5.2.6)). From the inductive hypothesis, we have that derivation (2.5.8) is of the form

(5.2.9) $\$S\$ ==>_{rm}^{*} \$\alpha'\delta\omega'\$ ==>_{rm}^{n-1} \$\alpha'\delta_{n-1}\omega'\$$;

Additionally, it follows from derivation (5.2.4) that $\delta_{n-1} ==> \beta$, and therefore, the following derivation exist in G:

(5.2.10) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta\omega'\$ ==>^{n-1}_{rm} \$\alpha'\delta_{n-1}\omega'\$ ==> \$\alpha'\beta\omega'\$.$

This derivation may be written as

(5.2.11) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta\omega'\$ ==>^{n}_{rm} \$\alpha'\beta\omega'\$.$

Derivation (5.2.11) is of the required form and the proof is complete.

Lemma 5.3 generalizes Lemma 5.2 to derivations having different numbers of steps. In this lemma, we establish that if $\alpha\beta\omega$ and $\alpha'\beta\omega'$ are derived from $\alpha\delta\omega$ and $\alpha'\delta'\omega'$, respectively, with $\delta' ==>^{+}_{rm} \beta$ being longer than $\delta ==>^{*}_{rm} \beta$, then $\delta' ==>^{*}_{rm} \delta ==>^{*}_{rm} \beta$, provided that $PREF_1(\omega)=PREF_1(\omega')$ and the last steps in both derivations are identical. Moreover, we show that if the last steps of these derivations are not identical, then, at least in one of the derivations, a unique symbol is erased in the last step.

LEMMA 5.3 Let G=(N,Σ,P,S) be an ESP grammar in which there exists derivations

(5.3.1) $\$S\$ ==>^{*}_{rm} \$\alpha\delta\omega\$ ==>^{n}_{rm} \$\alpha\beta\omega\$;$

(5.3.2) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta'\omega'\$ ==>^{m}_{rm} \$\alpha'\beta\omega'\$;$

such that

(1) $PREF_1(\omega\$)=PREF_1(\omega'\$)$;

(2) $\$\alpha' <+ \delta'$;

Then, either derivation (5.3.2) is of the form

(5.3.3) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta'\omega'\$ ==>^{m-n}_{rm} \$\alpha'\delta\omega'\$ ==>^{n}_{rm} \$\alpha'\beta\omega'\$;$

or,

(5.3.4) There exist ϑ and ϑ' in V_G^+ such that

$$\delta ==>^{*}_{rm} \vartheta ==> \beta;$$

$$\delta' ==>^{*}_{rm} \vartheta' ==> \beta;$$

where $PREF_1(\vartheta) \neq PREF_1(\vartheta')$, and there exists Z in N_ε(G) such that either $\vartheta = Z\beta$ or $\vartheta' = Z\beta$.

Proof. The proof proceeds by induction on n.

BASIS: $n=0$. If $n=0$, we have that $\delta=\beta$, and derivation (5.3.2) is of the form required by derivation (5.3.3).

INDUCTIVE STEP: Assume that the result holds for all k, $k<n$, $n>0$. Then, derivation (5.3.1) may be written as

(5.3.5) $\$S\$ ==>^{*}_{rm} \$\alpha\delta\omega\$ ==>^{n-1}_{rm} \$\alpha\delta_{n-1}\omega\$ ==>_{rm} \$\alpha\beta\omega\$.$

Also, since n is assumed to be greater than zero, and $m \geq n$, by applying Lemma 3.6 to derivation (5.3.2) we have that α' has only $\lessdot$ and $\doteq$. Hence, derivations (5.3.5) and (5.3.2) satisfy conditions (1) ($PREF_1(\omega\$) = PREF_1(\omega'\$)$) and (3) ($\$\alpha'$ has only $\lessdot$ and $\doteq$) of Lemma 5.1. Now if condition (2) (either $\$\alpha' \lessdot \beta$ and ($\$\alpha \lessdot \beta$ or $\$\alpha \doteq \beta$) or $SUFF_1(\$\alpha) = SUFF_1(\$\alpha')$) is also satisfied, then we can apply Lemma 5.1 to derivations (5.3.5) and (5.3.2) written as

(5.3.6) $\$S\$ ==>^{*}_{rm} \$\alpha\delta_{n-1}\omega\$ ==> \$\alpha\beta\omega\$,$

(5.3.7) $\$S\$ ==>^{+}_{rm} \$\alpha'\beta\omega'\$$

and conclude that derivation (5.3.7) is of the form

(5.3.8) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta_{n-1}\omega'\$ ==>_{rm} \$\alpha'\beta\omega'\$.$

If condition (2) of Lemma 5.1 is not satisfied, however, we must have that $\$\alpha' \lessdot \beta$ and ($\$\alpha \lessdot \beta$ or $\$\alpha \doteq \beta$) or $SUFF_1(\$\alpha) = SUFF_1(\$\alpha')$ is not true. This implies ($\$\alpha \gtrdot \beta$ or $\$\alpha' \gtrdot \beta$) and $SUFF_1(\$\alpha) \neq SUFF_1(\$\alpha')$ (Observe since $\alpha' \lessdot \delta'$ and G has no precedence conflicts $\alpha' \doteq \beta$ is not possible). Now consider

$\alpha' \lessdot \delta'$ and G has no precedence conflicts $\alpha' \stackrel{\pm}{=} \beta$ is not possible). Now consider derivations (5.3.6) and

(5.3.9) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta'_{m-1}\omega'\$ ==>_{rm} \$\alpha'\beta\omega'\$.$

Since either $\$\alpha \doteq> \beta$, or $\$\alpha' \doteq> \beta$, we have that either $\delta_{n-1} = Z\beta$, or $\delta'_{m-1} = Z'\beta$, for some Z, $Z' \in N_\varepsilon(G) \cup \{\varepsilon\}$. If $Z = Z'$, then $\delta_{m-1} = \delta_{n-1}$ and derivation (5.3.9) is identical to derivation (5.3.8). Otherwise, we have shown that

$$\delta ==>^{*}_{rm} \delta_{n-1} ==> \beta, \text{ and}$$
$$\delta' ==>^{*}_{rm} \delta'_{m-1} ==> \beta,$$

where $PREF_1(\delta_{n-1}) \neq PREF_1(\delta'_{m-1})$, and either δ_{n-1} or δ'_{m-1} starts with a symbol in $N_\varepsilon(G)$. This implies that conclusion (5.3.4) holds. To complete the proof, we need to show that if derivation (5.3.2) is of the form shown by derivation (5.3.8), then it is of the form required by derivation (5.3.3).

Derivation (5.3.8) may be rewritten as:

(5.3.10) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta'\omega'\$ ==>^{m-1}_{rm} \$\alpha'\delta_{n-1}\omega'\$ ==>_{rm} \$\alpha'\beta\omega'\$.$

Applying the induction hypothesis to derivations (5.3.5) and (5.3.10), we conclude that (5.3.10) is of the form

(5.3.11) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta'\omega'\$ ==>^{(m-1)-(n-1)}_{rm} \$\alpha'\delta\omega'\$ ==>^{n-1}_{rm}$
$\$\alpha'\delta_{n-1}\omega'\$ ==>_{rm} \$\alpha'\beta\omega'\$$

That is, derivation (5.3.2) is of the form

$$\$S\$ ==>^{*}_{rm} \$\alpha'\delta'\omega'\$ ==>^{m-n}_{rm} \$\alpha'\delta\omega'\$ ==>^{n}_{rm} \$\alpha'\beta\omega'\$;$$

and we have the established the desired result.

The next lemma shows how two derivations may be interleaved in an ESP grammar. Additionally, this lemma demonstrates how ε-rules can be used to prevent this type of interleaving.

LEMMA 5.4 Let G=(N,Σ,P,S) be an ESP grammar, $\alpha_1, \alpha_2, \beta_1, \beta_2 \in V_G^*$, and $u_1, u_2, v_1, v_2, \omega_1, \omega_2, x_1, x_2, y_1, y_2 \in \Sigma^*$. Suppose $v_1, v_2 \in a^+$, for some $a \in \Sigma^+$, and the following derivations exist in G:

$\$S\$ ==>_{rm}^* \alpha_1 A_1 y_1$; $A_1 ==>_{rm}^* \beta_1 A_1 x_1$; $A_1 ==>_{rm}^* \omega_1$; $\alpha_1 ==>_{rm}^* u_1$; $\beta_1 ==>_{rm}^* v_1$;

$\$S\$ ==>_{rm}^* \alpha_2 A_2 y_2$; $A_2 ==>_{rm}^* \beta_2 A_2 x_2$; $A_2 ==>_{rm}^* \omega_2$; $\alpha_2 ==>_{rm}^* u_2$; $\beta_2 ==>_{rm}^* v_2$.

Then either

(*i*) For all $k \geq 0$, there exist $z \in \Sigma^+$, such that $PREF_1(z) = PREF_1(a)$, and

(5.4.1) $\$S\$ ==>_{rm}^* \$\alpha_1 \beta_1^k \beta_2 z\$ ==>_{rm}^* \$\alpha_1 \beta_1^k v_2 z\$$;

or,

(*ii*) There exist ϑ and ϑ' in V_G^+ such that

$$\beta_1 ==>_{rm}^* \vartheta ==> v_1;$$

$$\beta_2 ==>_{rm}^* \vartheta' ==> v_2;$$

where $PREF_1(\vartheta) \neq PREF_1(\vartheta')$, and there exists Z in N_ε(G) such that either $\vartheta = Z v_1$ or $\vartheta' = Z v_2$.

Proof. Let $\beta_1 ==>_{rm}^{n_1} v_1 = a^i$, and $\beta_2 ==>_{rm}^{n_2} v_2 = a^j$, for some $n_1, n_2 \geq 0$, and $i, j > 0$. We first show that if $n_1 = 0$, then $n_2 = 0$. Assume, for the sake of contradiction, that $n_1 = 0$, but $n_2 > 0$. That is, $\beta_1 = v_1$, and $\beta_2 = \gamma B v'_2 ==>_{rm}^+ v''_2 v'_2$, $\gamma \in V_G^*$, $B \in N$, and v_2'', $v_2' \in \Sigma^*$. Now consider the following derivation

(5.4.2) $\$S\$ ==>_{rm}^* \$\alpha_1 \beta_1 A_1 x_1 y_1 \$ = \$\alpha_1 v_1 A_1 x_1 y_1 \$ ==>_{rm}^+ \$\alpha_1 v_1 v_1 A_1 x_1 x_1 y_1 \$$.

Applying Lemma 3.6 to derivation (5.4.2), we conclude that v_1 has only $\lessdot$ and $\doteq$, and $v_1 \lessdot v_1$. But $v_1 = a^i$; hence a has only $\lessdot$ and $\doteq$ and $a \lessdot a$. Now consider the derivation

(5.4.3) $\$S\$ ==>^{*}_{rm} \$\alpha_2\beta_2 A_2 x_2 y_2\$ ==>^{*}_{rm} \$\alpha_2\beta_2\beta_2 A_2 x_2 x_2 y_2\$ ==>^{*}_{rm}$

$\$\alpha_2\beta_2\beta_2\omega_2 x_2 x_2 y_2\$ = \$\alpha_2\gamma B v'_2\gamma B v'_2\omega_2 x_2 x_2 y_2\$ ==>^{+}_{rm}$

$\$\alpha_2\gamma B v'_2 v''_2 v'_2\omega_2 x_2 x_2 y_2\$ ==>^{+}_{rm}$

$\$\alpha_2 v''_2 v'_2 v''_2 v'_2\omega_2 x_2 x_2 y_2\$.$

Obviously, $v_2' v_2' = v_2 = a^j$ is not the empty string. We consider three different cases and in each case derive a contradiction to the fact that a has only $\not\Leftarrow$ and $\stackrel{+}{=}$ and $a \not\Leftarrow a$.

CASE 1 $v''_2 \neq \varepsilon$ and $v_2' \neq \varepsilon$.

Applying Lemma 3.6 to derivation (5.4.3), we get that $v''_2 \not\Rightarrow v'_2$, and we have a contradiction to a having only $\not\Leftarrow$ and $\stackrel{+}{=}$, and $a \not\Leftarrow a$.

CASE 2 $v''_2 = \varepsilon$ and $v'_2 \neq \varepsilon$.

Applying Lemma 3.6 to derivation (5.4.3), we get that $v'_2 \not\Rightarrow v'_2$, but, $v'_2 = a^j$. Hence $a \not\Rightarrow a$, and we have a contradiction to $a \not\Leftarrow a$.

CASE 3 $v''_2 \neq \varepsilon$ and $v'_2 = \varepsilon$.

Again applying Lemma 3.6 to derivation (5.4.3), we conclude that $v''_2 \not\Rightarrow v''_2$, which similar to the previous case implies $a \not\Rightarrow a$, a contradiction to $a \not\Leftarrow a$.

From the arguments given above, we conclude that if $n_1 = 0$, then $n_2 = 0$.

Now, choose m, l such that $m \times n_1 \geq n_2$, and $(l+1) \times j > m \times i > j$. Then, for each $k \geq 0$, the following pair of derivations exist in G.

(5.4.4) $\$S\$ ==>^{*}_{rm} \$\alpha_2\beta_2 v_2^{l} \omega_2 x_2^{l+1} y_2 \$ ==>^{n2}_{rm} \$\alpha_2 v_2 v_2^{l} \omega_2 x_2^{l+1} y_2 \$;$

(5.4.5) $\$S\$ ==>^{*}_{rm} \$\alpha_1 \beta_1^{k} \beta_1^{m} v_1 \omega_1 x_1^{k+m+1} y_1 \$ ==>^{m \times n_1}_{rm}$

$\$\alpha_1 \beta_1^{k} v_1^{m} v_1 \omega_1 x_1^{k+m+1} y_1 \$.$

Let $\beta = v_1^{m}$, $\delta' = \beta_1^{m}$, $\delta = \beta_2 \alpha^{i \times m - j}$, $\alpha' = \alpha_1 \beta_1^{k}$, $\alpha = \alpha_2$, $\omega' = v_1 \omega_1 x_1^{k+m+1} y_1$, and $\omega = \alpha^{(l+1) \times j - i \times m} \omega_2 x_2^{l+1} y_2$. With this notation, derivations (5.4.4) and (5.4.5) can be rewritten as follows:

(5.4.6) $\$S\$ ==>^{*}_{rm} \$\alpha\delta\omega\$ ==>^{n2}_{rm} \$\alpha\beta\omega\$;$

(5.4.7) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta'\omega'\$ ==>^{m \times n 1}_{rm} \$\alpha'\beta\omega'\$.$

All the conditions of Lemma 5.3 are satisfied. Thus, either derivation (5.4.7) is of the form

(5.4.8) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta'\omega'\$ ==>^{m \times n 1 - n 2}_{rm} \$\alpha'\delta\omega'\$ ==>^{n2}_{rm} \$\alpha'\beta\omega'\$,$

or, there exist ϑ and ϑ' such that

$$\delta ==>^{*}_{rm} \vartheta ==> \beta,$$

$$\delta' ==>^{*}_{rm} \vartheta' ==> \beta,$$

where either $PREF_1(\vartheta) \neq PREF_1(\vartheta')$, and there exists Z in $N_\varepsilon(G)$ such that either $\vartheta = Z\beta$ or $\vartheta' = Z\beta$.

If the latter conclusion holds, we observe that $\delta = \beta_2 \alpha^{i \times m - j}$, and $\delta' = \beta_1^{m}$, and conclude that that conclusion (ii) of the lemma holds. On the other hand, if the first conclusion above holds, we may choose $z = \alpha^{i \times m - j} \omega'$ and write derivation (5.4.8) as

(5.4.9) $\$S\$ ==>^{*}_{rm} \$\alpha'\delta\omega' = \alpha_1 \beta_1^{k} \beta_2 z \$ ==>^{n2} \alpha_1 \beta_1^{k} v_2 z \$,$

and we have shown that conclusion (i) holds.

We now present the iteration theorem for ESP_i languages.

THEOREM 5.5 Let L=L(G) be an ESP_i language, where G=(N,Σ,P,S) is an ESP_i grammar. Assume that for all j, $1 \le j \le i+2$, there exists $s_j \in$L, with G-factorization $u_j v_j w_j x_j y_j$ such that $v_j \in a^+$, for some $a \in \Sigma^+$, and for all k, $j < k \le i+2$, there exist $r_k \ge 0$, and $z_k \in \Sigma^+$, such that

(1) $PREF_1(z_k) = PREF_1(x_k y_k)$;

(2) $u_j v_j^{r_k} v_k^2 w_k x_k z_k \in L$.

Then, there exist p, q, $1 \le p < q \le i+2$, such that for all $m \ge 0$, $u_p v_p^{r_q} v_q^{m+1} w_q x_q^m z_q \in L$.

Proof. For all $1 \le j \le i+2$, $s_j \in L$ with G-factorization $u_j v_j w_j x_j y_j$ implies that the following derivations exist in G (see Theorem 2.21)

$\$S\$ ==>^*_{rm} \alpha_j A_j y_j$; $A_j ==>^*_{rm} \beta_j A_j x_j$; $A_j ==>^*_{rm} w_j$; $\alpha_j ==>^*_{rm} u_j$; $\beta_j ==>^*_{rm} v_j$.

Moreover, for each pair of integers j,k, $1 \le j < k \le i+2$, we can apply Lemma 5.4 to the derivations induced by G-factorizations of s_j and s_k and conclude that either (5.5.1) or (5.5.2) holds.

(5.5.1) for all $n \ge 0$, there exist $z'_k \in \Sigma^+$, such that $PREF_1(z'_k) = PREF_1(a)$, and

$\$S\$ ==>^*_{rm} \$ \alpha_j \beta_j^n \beta_k z'_k \$ ==>^*_{rm} \$ \alpha_j \beta_j^n v_k z'_k \$$,

(5.5.2) there exist ϑ_j and ϑ_k in V_G^+ such that

$$\beta_j ==>^*_{rm} \vartheta_j ==> v_j;$$

$$\beta_k ==>^*_{rm} \vartheta_k ==> v_k.$$

where $PREF_1(\vartheta_j) \ne PREF_1(\vartheta_k)$, and there exists B in N_ε(G) such that either $\vartheta_j = Bv_j$, or $\vartheta_k = Bv_k$.

But, $N_\varepsilon(G)\cup\{\varepsilon\}$ has exactly $i+1$ members and it is not possible for conclusion (5.5.2) to hold for every pair of integers j,k, $1\le j<k\le i+2$. Thus, we can find p,q, $1\le p<q\le i+2$, such that by applying Lemma 5.4 to the derivations induced by G-factorizations of s_p and s_q, we may conclude that there exists $z'_q\in\Sigma^+$, such that $PREF_1(z_q)=PREF_1(\alpha)$, and for all $n\ge 0$ the following derivations exists in G

(5.5.3) $\$S\$ ==>^*_{rm} \$\alpha_p\beta_p^n\beta_q z'_q\$ ==>^*_{rm} \$\alpha_p\beta_p^n v_q z'_q\$.$

Furthermore, from derivations induced by G-factorization of s_q and assumption (2) of the lemma it follows that the following derivations exist in G.

(5.5.4) $\$S\$ ==>^*_{rm} \$\alpha_p\beta_p^{r_q} v_p w_p x_p^{r_q+1} y_p ==>^*_{rm} u_p v_p^{r_q} v^p w_p x_p^{r_q+1} y_p\$,$

(5.5.5) $\$S\$ ==>^*_{rm} \$u_p v_p^{r_q} v_q^2 w_q x_q z_q.$

Let $\alpha=\alpha'=\varepsilon$, $\beta=u_p v_p^{r_q}$, $\delta=\alpha_p\beta_p^{r_q}$, $w=v_p w_p x_p^{r_q+1} y_p$, and $w'=v_q^2 w_q x_q z_q$. With this notation, derivations (5.5.4) and (5.5.5) are of the following form

(5.5.6) $\$S\$ ==>^*_{rm} \$\alpha\delta w\$ ==>^*_{rm} \$\alpha\beta w\$,$

(5.5.7) $\$S\$ ==>^*_{rm} \$\alpha'\beta w'\$,$

and Lemma 5.2 applies. Hence, derivation (5.5.7) is of the form

(5.5.8) $\$S\$ ==>^*_{rm} \$\alpha'\delta w'\$ ==>^*_{rm} \$\alpha'\beta w'\$.$

That is,

(5.5.9) $\$S\$ ==>^*_{rm} \$\alpha_p\beta_p^{r_q} v_q^2 w_q x_q z_q\$ ==>^*_{rm} \$u_p v_p^{r_q} v_q^2 w_q x_q z_q\$.$

Now set n of the derivation (5.5.3) to r_q and apply Lemma 5.2 to derivations (5.5.3) and (5.5.9) with $\alpha=\alpha'=\alpha_p\beta_p^{r_q}$, $\beta=v_q$, $\delta=\beta_q$, $w=z'_q$, and $w'=v_q w_q x_q z_q$, and conclude that derivation (5.5.9) is of the form

(5.5.10) $\$S\$ ==>^{*}_{rm} \$\alpha_p \beta_p^{\tau_q} \beta_q v_q w_q x_q z_q \$ ==>^{*}_{rm} \$\alpha_p \beta_p^{\tau_q} v_q v_q w_q x_q z_q \$ =$

$$\$\alpha_p \beta_p^{\tau_q} v_q^2 w_q x_q z_q \$.$$

Moreover, it follows from the derivations induced by G-factorization of s_q that the following derivation exists in G

(5.5.11) $\$S\$ ==>^{*}_{rm} \$\alpha_q \beta_q A_q x_q y_q \$ ==>^{*}_{rm} \alpha_q \beta_q v_q w_q x_q^2 y_q \$.$

Again, we can apply Lemma 5.4 to derivation (5.5.11) and derivation (5.5.10) written as

(5.5.12) $\$S\$ ==>^{*}_{rm} \$\alpha_p \beta_p^{\tau_q} \beta_q v_q w_q x_q z_q \$,$

with notation $\beta=v_q w_q x_q$, $\delta=A_q$, $\alpha=\alpha_q \beta_q$, $\alpha'=\alpha_q \beta_q^{\tau_q} \beta_q$, $w=x_q y_q$, and $w'=z_q$, to conclude that derivation (5.5.12) is of the form

(5.5.13) $\$S\$ ==>^{*}_{rm} \$\alpha_p \beta_p^{\tau_q} \beta_q A_q z_q \$ ==>^{*}_{rm} \$\alpha_p \beta_q^{\tau_q} \beta_q v_q w_q x_q z_q \$.$

But $A_q ==>^{*}_{rm} v_q A_q x_q$ and $A_q ==>^{*} w_q$. Hence, for all $m \geq 0$, we have that

(5.5.14) $\$S\$ ==>^{*}_{rm} \$\alpha_p \beta_p^{\tau_q} \beta_q A_q z_q \$ ==>^{*}_{rm} \$\alpha_p \beta_p^{\tau_q} \beta_q v_q^m w_q x_q^m z_q \$ ==>^{*}_{rm}$

$$\$u_p v_p^{\tau_q} v_q^{m+1} w_q x_q^m z_q \$.$$

That is, for all $m \geq 0$, $u_p v_p^{\tau_q} v_q^{m+1} w_q x_q^m z_q \in L$.

5.2 The Hierarchy of ESP_i Languages

In Chapter 3, we showed that the class of simple precedence languages was properly included in the class of ESP_1 languages. This section extends this result by showing that the descriptive power of ESP languages is directly related to the number of ε-rules allowed. That is, for all i, $i \geq 0$, the class of ESP_i languages is properly included in the class of ESP_{i+1} languages. To establish this result, we prove that for all i, $i \geq 0$, the language $I_i = \bigcup_{j=1}^{i+2} \{a_j 0^n 1^{j \times n} \mid n \geq 1\}$

is ESP_{i+1} but not ESP_i.

THEOREM 5.6 For all $i \geq 0$, the class of ESP_i languages is properly included in the class of ESP_{i+1} languages.

Proof. Let $L_i = \bigcup_{j=1}^{i+2} \{a_j 0^n 1^{j \times n} \mid n \geq 1\}$. Consider the grammar G_i shown below.

G_i:

$S \rightarrow a_j X_j$	$1 \leq j \leq i+2$
$X_j \rightarrow A_j \; X_j \; C_j 1$	$2 \leq j \leq i+2$
$X_j \rightarrow A_j \; C_j 1$	$2 \leq j \leq i+2$
$A_j \rightarrow Z_j \; 0$	$2 \leq j \leq i+2$
$C_j \rightarrow C_{j-1} 1$	$3 \leq j \leq i+2$
$C_2 \rightarrow 1$	
$Z_j \rightarrow \varepsilon$	$2 \leq j \leq i+2$
$X_1 \rightarrow A_1 X_1 1$	
$X_1 \rightarrow A_1 1$	
$A_1 \rightarrow 0$	

The sets **L**, $\mathbf{L}_\varepsilon$, **R**, and $\mathbf{R}_\varepsilon$ for G_i are shown in Table 4. Table 5 shows the EP relations for G_i.

	L	L_ε	R	R_ε
S	a_j, a_1	ϕ	$X_j, X_1, 1$	ϕ
X_j	A_j, Z_j	0	1	ϕ
C_j	$C_{j-1}, \cdots, C_2, 1$	ϕ	1	ϕ
A_j	Z_j	0	0	ϕ
Z_j	ϕ	ϕ	ϕ	ϕ
X_1	$A_1, 0$	ϕ	1	ϕ
A_1	0	ϕ	0	ϕ

Table 4. **L**, $\mathbf{L}_\varepsilon$, **R**, $\mathbf{R}_\varepsilon$ for G_i, $2 \le j \le i+2$.

	S	X_j	C_j	A_j	Z_j	C_k	X_1	A_1	a_j	a_1	0	1	\$
S													
X_j			$\doteq$			$\lessdot$						$\lessdot$	$\gtrdot$
C_j												$\doteq$	
A_j		$\doteq$	$\doteq$	$\lessdot$	$\lessdot$	$\lessdot$					$\gtrdot$	$\lessdot$	
Z_j											$\doteq$		
X_1												$\doteq$	$\gtrdot$
C_k												$\doteq$	
A_1							$\doteq$	$\lessdot$			$\lessdot$	$\doteq$	
a_j		$\doteq$		$\lessdot$	$\lessdot$						$\gtrdot$		
a_1							$\doteq$	$\lessdot$			$\lessdot$		
0											$\gtrdot$	$\gtrdot$	
1												$\gtrdot$	$\gtrdot$
\$									$\lessdot$	$\lessdot$			

Table 5. The EP relations for G_i, $2 \le j \le i+1$, $2 \le k < j$.

Obviously, $L_i = L(G_i)$, G_i is AUI, EP relations for G_i are pairwise disjoint, and G_i has $i+1$ ε-rules. Moreover, for all j, $2 \le j \le i+2$,

$$\rho(Z_j) = \{(a_j, 0), (A_j, 0)\},$$

and we have that for all j, k, $2 \le j$, $k \le i+2$, $j \ne k$, $\rho(Z_j) \cap \rho(Z_k) = \phi$. Additionally, for all j, $2 \le j \le i+2$, the rightmost symbol of no production in G_i is related to Z_j and $(S, \$) \notin \rho(Z_j)$. Hence, G_i is ESP_{i+1}, and L_i is an ESP_{i+1} language. To this

end, we show that L_i is not an ESP_i language.

Assume, for the sake of contradiction that for all $i \geq 0$, $L_i = L(G_i)$, where G_i is an ESP_i grammar. Let n be the integer from the iteration theorem for context-free languages, and for all j, $1 \leq j \leq i+2$, consider $s_j = a_j 0^n 1^{j \times n}$ in L_i. In each G-factorization $u_j v_j w_j x_j y_j$ of s_j, we must have that

$$u_j = a_j 0^{b_j},$$
$$v_j = 0^{c_j},$$
$$w_j = 0^{d_j} 1^{e_j},$$
$$x_j = 1^{j \times c_j},$$
$$y_j = 1^{f_j},$$

for some $c_j > 0$, and b_j, d_j, e_j, $f_j \geq 0$. Otherwise, we will have a violation for $u_j v_j w_j x_j y_j$ being a G-factorization for s_j. Now, for each pair of integers j, k, $1 \leq j < k \leq i+2$, choose r_k such that $j \times (b_j + r_k \times c_j + 2 \times c_k + d_k) > e_k + k \times c_k$, and let $z_k = 1^{j \times (b_j + r_k \times c_j + 2 \times c_k + d_k) - (e_k + k \times c_k)}$. Then, for all $j, k, 1 \leq j < k \leq i+2$, the string $u_j v_j^{r_k} v_k^2 w_k x_k z_k = a_j 0^{b_j} 0^{r_k \times c_j} 0^{2 \times c_k} 0^{d_k} 1^{e_k} 1^{k \times c_k} z_k = a_j 0^{b_j + r_k \times c_j + 2 \times c_k + d_k} 1^{j \times b_j + r_k \times c_j + 2 \times c_k + d_k}$ belongs to L_i. Moreover, for each j, and k, $1 \leq j < k \leq i+2$, v_j, $v_k \in 0^+$, x_k, y_k, and z_k belong to 1^+. Hence, all of the conditions of the iteration theorem for ESP_i languages are satisfied . Thus, there must exist p, q, $1 \leq p < q \leq k+2$, such that for all $m \geq 0$, $u_p v_p^{r_q} v_q^{m+1} w_q x_q^m z_q$ belongs to L. But, $u_p v_p^{r_q} v_q^{m+1} w_q x_q^m z_q$ is of the form

$$a_p \; 0^{b_p} \; 0^{r_q \times c_p} \; 0^{(m+1) \times c_q} \; 0^{d_q} 1^{e_q} \; 1^{m \times q \times c_q} z_q =$$
$$a_p 0^{b_p + r_q \times c_p + (m+1) \times c_q + d_q} \; 1^{e_q} \; 1^{m \times q \times c_q} \; 1^{p \times (b_p + r_q \times c_p + 2 \times c_q + d_q) - (e_q + q \times c_q)} =$$
$$a_p 0^{b_p + r_q \times c_p + (m+1) \times c_q + d_q} 1^{p \times (b_p + r_q \times c_p + 2 \times c_q + d_q) + (m-1) \times q \times c_q}.$$

Hence, for all $m \geq 0$, we must have

$$p \times (b_p + r_q \times c_p + (m+1) \times c_q + d_q) = p \times (b_p + r_q \times c_p + 2 \times c_q + d_q) + (m-1) \times q \times c_q .$$

That is, for all $m \geq 0$, we must have that

$$p \times (m+1) \times c_q = 2 \times p \times c_q + (m-1) \times q \times c_q .$$

But, for $m=2$, this implies that $p \times 3 \times c_q = 2 \times p \times c_q + q \times c_q$. That is, $p \times c_q = q \times c_q$. But, p, q, and c_q are all greater than zero. Hence, we must have $p = q$. This is contradictory to the assumption that $p > q$. Thus, L_i is not ESP_i.

The hierarchy of ESP languages is shown in Figure 7.

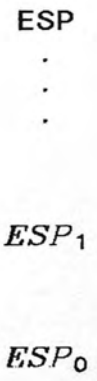


Figure 7. The hierarchy of epsilon simple precedence languages.

5.3 Closure Properties of ESP_i Languages

The closure properties of ESP_i languages under the operations union, intersection, and homomorphism are discussed in this section. We show that for each i, $i \geq 0$, the class of ESP_i languages is not closed under any of these operations.

LEMMA 5.7 For each i, $i \geq 0$, the class of ESP_i languages is not closed under intersection.

Proof. For each i, $i \geq 0$, consider the languages L_i, and L'_i defined below.

$$L_i = \bigcup_{j=1}^{i+1} \{a_j 0^n 1^{j \times n} \mid n \geq 1\} \cup \{x^k y^k z^j \mid k, j \geq 1\}$$

$$L'_i = \bigcup_{j=1}^{i+1} \{a_j 0^n 1^{j \times n} \mid n \geq 1\} \cup \{x^k y^j z^j \mid k, j \geq 1\}$$

Grammars G_i, and G'_i, shown below, are ESP_i grammars defining L_i, and L'_i, respectively. The EP relations for G_i and G'_i are shown in Tables 6 and 7.

G_i:

$S \rightarrow a_j X_j$ $\quad 1 \leq j \leq i+1$

$X_j \rightarrow A_j\ X_j\ C_j 1$ $\quad 2 \leq j \leq i+1$

$X_j \rightarrow A_j\ C_j 1$ $\quad 2 \leq j \leq i+1$

$A_j \rightarrow Z_j\ 0$ $\quad 2 \leq j \leq i+1$

$C_j \rightarrow C_{j-1} 1$ $\quad 3 \leq j \leq i+1$

$C_2 \rightarrow 1$

$Z_j \rightarrow \varepsilon$ $\quad 2 \leq j \leq i+1$

$X_1 \rightarrow A_1 X_1 1$

$X_1 \rightarrow A_1 1$

$A_1 \rightarrow 0$

$S \rightarrow E\ F$

$E \rightarrow x\ E\ y$

$E \rightarrow x\ y$

$F \rightarrow z\ F$

$F \rightarrow z$

G'_i:

$S \rightarrow a_j X_j$	$1 \leq j \leq i+1$
$X_j \rightarrow A_j\ X_j\ C_j 1$	$2 \leq j \leq i+1$
$X_j \rightarrow A_j\ C_j 1$	$2 \leq j \leq i+1$
$A_j \rightarrow Z_j\ 0$	$2 \leq j \leq i+1$
$C_j \rightarrow C_{j-1} 1$	$3 \leq j \leq i+1$
$C_2 \rightarrow 1$	
$Z_j \rightarrow \varepsilon$	$2 \leq j \leq i+1$
$X_1 \rightarrow A_1 X_1 1$	
$X_1 \rightarrow A_1 1$	
$A_1 \rightarrow 0$	
$S \rightarrow F\ E$	
$E \rightarrow y\ E\ z$	
$E \rightarrow y\ z$	
$F \rightarrow x\ F$	
$F \rightarrow x$	

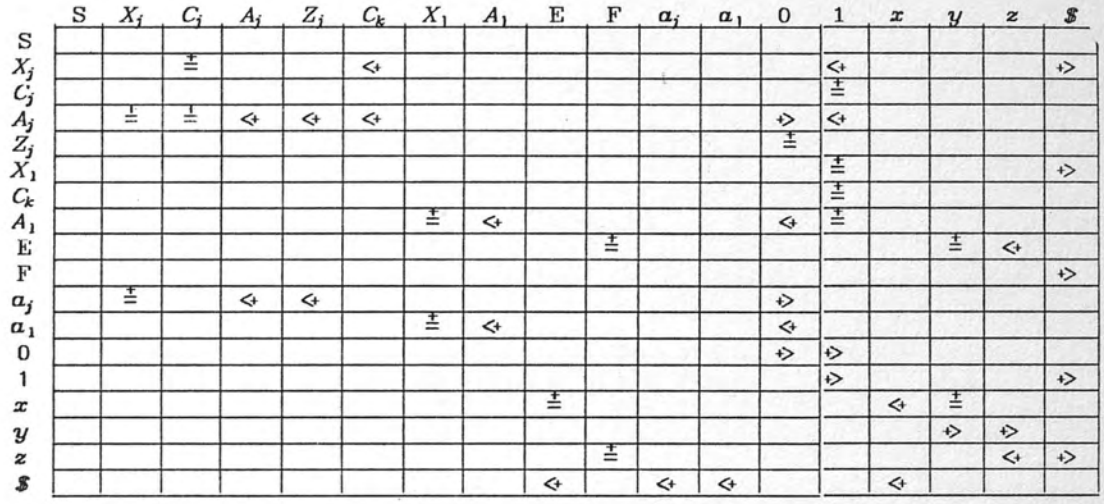


Table 6. The EP relations for G_i, $2 \leq j \leq i+1$, $2 \leq k < j$.

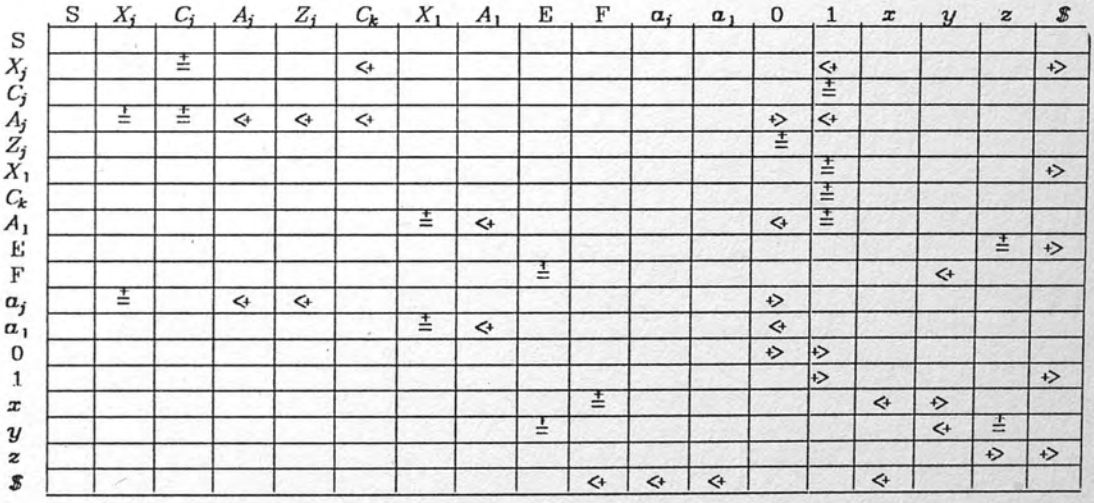


Table 7. The EP relations for G'_i, $2 \leq j \leq i+1$, $2 \leq k < j$.

Now consider the language

$$L_i \cap L'_i = \{x^k y^k z^k \mid k \geq 1\}.$$

This language is known not to be context-free [4]. Obviously, every ESP_i

language is context-free. Therefore, for all j, $j \geq 0$, $L_j \cap L'_j$ is not context-free, and we have the desired result.

LEMMA 5.8 For each i, $i \geq 0$, the class of ESP_i languages is not closed under union.

Proof. For each i, $i \geq 0$, consider the languages L_i, and L'_i defined below.

$$L_i = \{ \bigcup_{j=1}^{i+1} \{a_j 0^n 1^{j \times n} \mid n \geq 1\}$$

$$L'_i = \{ \bigcup_{j=i+2}^{2 \times i+2} \{a_j 0^n 1^{j \times n} \mid n \geq 1\}$$

Obviously, both L_i, and L'_i are ESP_i languages. Now consider

$$L_i \cup L'_i = \{ \bigcup_{j=1}^{2 \times i+2} \{a_j 0^n 1^{j \times n} \mid n \geq 1\}.$$

Using an argument similar to the one used in Theorem 5.6, we can show that $L_i \cup L'_i$ is at least $ESP_{2 \times i+1}$. Hence, the class of ESP_i languages are not closed under union.

LEMMA 5.9 For each i, $i \geq 0$, the class of ESP_i languages is not closed under homomorphism.

Proof. For each i, $i \geq 0$, consider the language L_i defined below.

$$L_i = \{ \bigcup_{j=1}^{i+1} \{a_j 0^n 1^{j \times n} \mid n \geq 1\} \cup \{a_{i+2} 0^n x^{(i+2) \times n} \mid n \geq 1\}.$$

Grammar G, shown below is an ESP_i grammar defining G.

G:

$S \rightarrow a_j X_j$ $\quad 1 \leq j \leq i+1$

$X_j \rightarrow A_j\ X_j\ C_j 1$ $\quad 2 \leq j \leq i+1$

$X_j \rightarrow A_j\ C_j 1$ $\quad 2 \leq j \leq i+1$

$A_j \rightarrow Z_j\ 0$ $\quad 2 \leq j \leq i+1$

$C_j \rightarrow C_{j-1} 1$ $\quad 3 \leq j \leq i+1$

$C_2 \rightarrow 1$

$Z_j \rightarrow \varepsilon$ $\quad 2 \leq j \leq i+1$

$X_1 \rightarrow A_1 X_1 1$

$X_1 \rightarrow A_1 1$

$A_1 \rightarrow 0$

$S \rightarrow a_{i+2} X_{i+2}$

$X_{i+2} \rightarrow A_1 X_{i+2} Y_{i+2} x$

$X_{i+2} \rightarrow A_1 Y_{i+2} x$

$Y_j \rightarrow Y_{j-1} x$ $\quad 2 \leq j \leq i+2$

$Y_2 \rightarrow x$

Define the homomorphism h as follows:

$h(a_j) = a_j$ $\quad 1 \leq j \leq i+2$

$h(0) = 0$

$h(1) = 1$

$h(x) = 1$

Obviously, $h(L_i) = \{ \bigcup_{j=1}^{i+2} \{a_j 0^n 1^{j \times n} \mid n \geq 1\}$, is not ESP_i. Hence, the class of ESP_i languages is not closed under homomorphism.

CHAPTER 6

PROPERTIES OF EPSILON PRECEDENCE GRAMMARS

In Chapter 3, we have shown that the class of ESP grammars are properly included in the class of EWP grammars. Also, in that chapter, we established that the classes of ESP and EWP languages are equivalent. Additionally, we proved in Chapter 4 that every deterministic context-free language is described by some EWP grammar.

The purpose of this chapter is to explore the inclusion relationships between ESP and EWP grammars and languages and the other known classes of context-free grammars and languages. We show that the classes of EWP and ESP grammars are properly included in the well-known classes of context-free grammars with ε-rules which describe exactly the class of deterministic context-free languages. The equivalence of ESP languages with the class of deterministic context-free languages in turn implies that the hierarchy of ESP languages established in Chapter 5 exhausts the class of deterministic context-free languages. This result is of particular importance since the hierarchies of LL(k) and strict deterministic languages cover only a subset of deterministic context-free languages.

We begin by generalizing the SR(s,k) concept [44] to obtain a larger class we shall call Extended SR(s,k), or ESR(s,k) for short. The class of ESR(s,k) grammars is then used as a vehicle to establish some desired properties. Recall from definition 2.14 that a state of an SR(s,k) parser is deterministic if there are no "shift-reduce" conflicts and no conflicts with "accept." "Reduce-reduce" con-

flicts, however, are permitted provided they can be resolved by looking "s" symbols below the shortest right-part among the conflicting productions. Our generalization allows for "shift-reduce" conflicts if they can be resolved in a manner similar to "reduce-reduce" conflicts.

DEFINITION 6.1 Let G=(N, Σ ,P, S) be a reduced context-free grammar. For $s,k \geq 0$, let $\mathbf{C}_{s,k}$ be the canonical collection of SR(s,k) states for G. Furthermore, let $\Lambda(\gamma) \in \mathbf{C}_{s,k}$ denote the SR(s,k) state defined by $\gamma \in V_G^{*\,s}$. Then G is said to be an ESR(s,k) grammar:

i) For $k=0$, if $[X \rightarrow \alpha.\beta], [Y \rightarrow \delta.] \in \Lambda(\gamma)$ then $\beta=\varepsilon$, and if $[X \rightarrow \alpha\beta.], [Y \rightarrow \beta.] \in \Lambda(\gamma)$ then not both $[X \rightarrow \alpha.\beta], [Y \rightarrow .\beta]$ belong to the same state.

ii) For $k>0$,

a) If $[Z \rightarrow \alpha S.\ \beta \mid v] \in \Lambda(SUFF_s(S))$ where $Z \neq S'$, then $\varepsilon \notin FIRST_k(\beta v)$ or $\alpha=\varepsilon$ and $[Z \rightarrow .S\beta \mid v] \notin \Lambda(\varepsilon)$

b) If $[A \rightarrow \alpha.\beta \mid u]$ and $[B \rightarrow \delta\alpha. \mid v]$ belong to $\Lambda(\gamma)$ where $v \in EFF_k(\beta u)$ then not both $[A \rightarrow .\alpha\beta \mid u]$ and $[B \rightarrow \delta.\alpha \mid v]$ belong to the same state.

c) If $[A \rightarrow \delta\alpha.\beta \mid u]$ and $[B \rightarrow \alpha. \mid v]$ belong to $\Lambda(\gamma)$ where $v \in EFF_k(\beta u)$ then not both $[A \rightarrow \delta.\alpha\beta \mid u]$ and $[B \rightarrow .\alpha \mid v]$ belong to the same state.

As our first result in this chapter, we establish that every ESR(m,n) grammar is also (m,n)BRC.

THEOREM 6.2 The class of ESR(s,k) grammars is included in the class of (s,k)BRC grammars.

Proof. We proceed by assuming G is not (s,k)BRC and then show G is not ESR(s,k). Suppose in the augmented grammar G', there are rightmost derivations:

$$S' ==>_{rm}^{*} \alpha A \omega ==>_{rm} \alpha\beta\omega$$

$$S' ==>_{rm}^{*} \gamma B x ==>_{rm} \gamma\delta x = \alpha'\beta y$$

where $y \in \Sigma^{*}$, $|x| \leq |y|$, $SUFF_s(\alpha') = SUFF_s(\alpha)$, $PREF_k(\omega) = PREF_k(y)$ and $B \rightarrow \delta \neq A \rightarrow \beta$. We must consider two cases.

CASE 1. $x = y$.

If $x = y$, then $\gamma\delta = \alpha'\beta$ and β is a suffix of δ or vice versa. In either case [$B \rightarrow \delta.$ | $PREF_k(y)$] and [$A \rightarrow \beta.$ | $PREF_k(y)$] belong to $\Lambda(SUFF_s(\gamma\delta))$; this follows because

$$\begin{aligned} &SUFF_s(\alpha\beta) = \\ &SUFF_s(SUFF_s(\alpha)\beta) = \\ &SUFF_s(SUFF_s(\alpha')\beta) = \\ &SUFF_s(\alpha'\beta). \end{aligned}$$

Suppose $\delta = \vartheta\beta$. Then

$$[B \rightarrow \vartheta.\beta \mid PREF_k(y)] \in \Lambda(SUFF_s(\gamma\vartheta)) = \Lambda(SUFF_s(\alpha')), \text{ and}$$

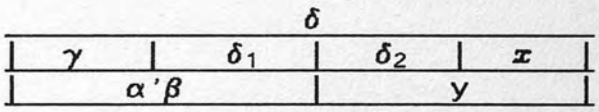
$$[A \rightarrow .\beta \mid PREF_k(y)] \in \Lambda(SUFF_s(\alpha)) = \Lambda(SUFF_s(\alpha')).$$

Clearly, if $k = 0$, condition (i) of Definition 6.1 is violated and if $k > 0$, condition (ii-c) of Definition 6.1 is violated. The argument assuming $\beta = \vartheta\delta$ is similar. Thus for the case $x = y$ we have shown G not to be ESR(s,k).

CASE 2. $|x|<|y|$.

In this case we consider two subcases:

a) $|x|<|y|\leq|\delta x|$



Assume $\delta=\delta_1\delta_2$, where $\delta_2\neq\varepsilon$ and $\gamma\delta_1=\alpha'\beta$; that is, $y=\delta_2 x$. It follows from the second derivation that $[B\rightarrow\delta_1.\delta_2 \mid PREF_k(x)]\in\Lambda(SUFF_s(\gamma\delta_1))=\Lambda(SUFF_s(\alpha'\beta))$. From the first derivation we also have $[A\rightarrow\beta. \mid PREF_k(\omega)]\in\Lambda(SUFF_s(\alpha\beta))=\Lambda(SUFF_s(\alpha'\beta))$. Now either $\beta=\gamma_2\delta_1$, where $\gamma=\gamma_1\gamma_2$ or $\delta_1=\delta'_1\beta$. If $\beta=\gamma_2\delta_1$, then we have the following:

$[B\rightarrow\delta_1.\delta_2 \mid PREF_k(x)]\in\Lambda(SUFF_s(\alpha'\beta))$,

$[A\rightarrow\gamma_2\delta_1. \mid PREF_k(\omega)]\in\Lambda(SUFF_s(\alpha'\beta))$,

$PREF_k(\omega)=EFF_k(\delta_2 PREF_k(x))$,

$[B\rightarrow.\delta_1\delta_2 \mid PREF_k(x)]\in\Lambda(SUFF_s(\gamma))$,

$[A\rightarrow\gamma_2.\delta_1 \mid PREF_k(\omega)]\in\Lambda(SUFF_s(\alpha\gamma_2))=\Lambda(SUFF_s(\alpha'\gamma_2))=$

$\Lambda(SUFF_s(\gamma_1\gamma_2))=$

$\Lambda(SUFF_s(\gamma))$.

Again, if $k=0$, condition (i) of Definition 6.1 is violated and if $k>0$, condition $(ii\text{-}b)$ of Definition 6.1 is violated. On the other hand, if $\delta_1=\delta'_1\beta$, then we have the following:

$[B\rightarrow\delta'_1\beta.\delta_2 \mid PREF_k(x)]\in\Lambda(SUFF_s(\alpha'\beta))$,

$[A\rightarrow\beta. \mid PREF_k(\omega)]\in\Lambda(SUFF_s(\alpha'\beta))$,

$PREF_k(\omega)=EFF_k(\delta_2 PREF_k(x))$,

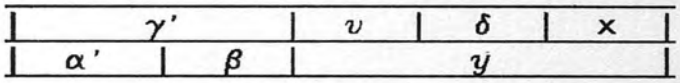
$[B\rightarrow\delta'_1.\beta\delta_2 \mid PREF_k(x)]\in\Lambda(SUFF_s(\gamma\delta'_1))=\Lambda(SUFF_s(\alpha'))=$

$\Lambda(SUFF_s(\alpha))$, and

$[A\rightarrow.\beta \mid PREF_k(\omega)]\in\Lambda(SUFF_s(\alpha))$.

Similar to the previous case, if k=0, condition (i) of Definition 6.1 is violated and if k>0, condition (ii-c) of Definition 6.1 is violated. Therefore, in this case, we have shown that G is not (s,k)BRC.

b) $|x|<|y|>|\delta x|$



Let $y = \upsilon\,\delta x$ for some $\upsilon \in \Sigma^+$. The second derivation can be written as:

$S' ==>^*_{rm} \gamma' Z x' ==>_{rm} \gamma' \vartheta_1 \vartheta_2 x' = \alpha' \beta \vartheta_2 x' ==>^*_{rm} \alpha' \beta \upsilon B x ==> \alpha' \beta \upsilon\,\delta x = \alpha' \beta y$,

br where $Z \rightarrow \vartheta_1 \vartheta_2$ is the first production in the rightmost derivation to form the substring $\alpha'\beta$. Now by identifying $Z \rightarrow \vartheta_1 \vartheta_2$ with the production $B \rightarrow \delta$ and recalling that $PREF_k(\omega) = PREF_k(y)$, where $\vartheta_2 x' ==>^*_{rm} y \in \Sigma^*$, an argument similar to that in part (a) can be applied to arrive at a violation to the conditions of Definition 6.1. Hence, G is not ESR(s,k).

In each case, we have shown that if G is not (s,k)BRC, then it is not ESR(s,k). Hence, we conclude that if G is (s,k)BRC, then it is ESR(s,k). That is, the class of ESR(s,k) grammars is included in the class of (s,k)BRC.

COROLLARY. If G is ESR(s,k), then G is not ambiguous.

Proof. This is a direct result of the fact that BRC grammars are unambiguous and every ESR(s,k) grammar is also (s,k)BRC.

We now show that the inclusion of ESR(s,k) grammars in the class of (s,k)BRC is proper. That is, not every (s,k)BRC grammar is also ESR(s,k). Consider the grammar G with the following productions.

S→aAc
S→bbB
A→bbc
B→bcc

The canonical collection of SR(1,1) states for G is shown below.

Λ(ε)	Λ(b)	Λ(a)
[S'→.S|ε]	[S→b.bB|ε]	[S→a.Ac|ε]
[S→.aAc|ε]	[S→bb.B|ε]	[A→.bbB|c]
[S→.bbB|ε]	[B→.bcc|ε]	
	[B→b.cc|ε]	
	[A→b.bc|c]	Λ(A)
Λ(S)	[A→bb.c|c]	[S→aA.c|ε]
[S'→S|ε]		
	Λ(c)	
Λ(B)	[B→bc.c|ε]	
[S→bbB.|ε]	[B→bcc.|ε]	
	[S→aAc.|ε]	
	[A→bbc.|c]	

In Λ(c) the items [B→bc.c|ε] and [A→bbc.|c] produce a "shift-reduce" conflict, and since the items [B→.bcc|ε] and [A→b.bc|c] both belong to Λ(b), G is not ESR(1,1). However, G is (1,1)BRC since the left context of A is "a" and the left context of B is "bb." Only one symbol below the handle is required to resolve the conflict when "bbc" is on top of the stack.

We now turn to exploring the relationship of ESR(1,1) grammars to EWP grammars. The key to our analysis is a result presented in lemma 6.3 which relates the EP relations $\lessdot$, $\doteq$, and $\gtrdot$ defined for a pair of symbols (X,Y) to the items existing in the SR state $\Lambda(X)$.

LEMMA 6.3 Let G=(N, Σ ,P, S) be an EWP grammar. Let $\mathbf{C}_{1,1}$ be the canonical collection of all SR(1,1) states for the augmented grammar G'.

6.3.1 The item [X→.|ε]∈Λ(ε) if and only if \$ $\gtrdot$ \$.

Proof. If [X→.|ε]∈Λ(ε), then either X=S or $S \Rightarrow^{+}_{rm} X\alpha \Rightarrow^{*}_{rm} \varepsilon$. In either case

$\varepsilon \in L(G)$ and it follows from the definition of EP relations that \$ $\Rightarrow$ \$. If \$ $\Rightarrow$ \$, then $\varepsilon \in L(G)$. Therefore, $S ==>^{*}_{rm} X ==>_{rm} \varepsilon$, for some $X \in N$. It follows from definition of $\Lambda(\varepsilon)$ that $[X \rightarrow . \mid \varepsilon] \in \Lambda(\varepsilon)$.

6.3.2 The item $[X \rightarrow . \mid v] \in \Lambda(\varepsilon)$, $v \neq \varepsilon$ if and only if \$ $\Rightarrow$ v.

Proof. If $[X \rightarrow . \mid v] \in \Lambda(\varepsilon)$, $v \neq \varepsilon$ then the derivation $S' ==> S ==>^{+}_{rm} X\alpha ==>^{*}_{rm} Xv\omega ==> v\omega$ must exist in the augmented grammar, G'. It follows that v belongs to $L_{\varepsilon}(S)$ and that \$ $\Rightarrow$ v. If \$ $\Rightarrow$ v, then from the definition of EP relations we have $v \in L_{\varepsilon}(S)$. Thus there is a production $\rho : X \rightarrow \varepsilon$ and leftmost derivation $\pi_1 \rho \pi_2$ such that $S ==>^{\pi 1}_{lm} X\vartheta ==>^{\rho}_{lm} \vartheta ==>^{\pi 2}_{lm} v\omega \in \Sigma^{+}$. By rearranging $\pi_1 \rho \pi_2$ we obtain a rightmost derivation, $\pi'_1 \rho \pi'_2$ such that $S ==>^{\pi' 1}_{rm} \alpha X v\omega ==>^{\rho}_{rm} \alpha v\omega ==>^{\pi' 2}_{rm} v\omega$. Now if $\alpha = \varepsilon$, then it follows from definition of $\Lambda(\varepsilon)$ that $[X \rightarrow . \mid v] \in \Lambda(\varepsilon)$. If $\alpha \neq \varepsilon$, then $\alpha \in N^{+}$, $\pi'_2 = \pi'' q$, $q : Z \rightarrow \varepsilon \subset P$ and $S' ==>^{\pi' 1 \rho}_{rm} \alpha v\omega ==>^{\pi''}_{rm} Zv\omega ==>^{q}_{rm} v\omega$. In this case $[Z \rightarrow . \mid v] \in \Lambda(\varepsilon)$.

6.3.3 The item $[X \rightarrow \alpha . \mid \varepsilon] \in \Lambda(\gamma)$, $\gamma \neq \varepsilon$, if and only if γ $\Rightarrow$ \$.

Proof. If $[X \rightarrow \alpha . \mid \varepsilon] \in \Lambda(\gamma)$, $\gamma \neq \varepsilon$, then either $\gamma = SUFF_1(\alpha)$ or $\alpha = \varepsilon$ and there is an item $I = [C \rightarrow \vartheta\gamma . A\lambda \mid \varepsilon]$ in $\Lambda(\gamma)$ such that $[X \rightarrow . \mid \varepsilon] \in$ closure (I) and $A\lambda ==>^{*}_{rm} \varepsilon$. In either case from the definition of $\Lambda(\gamma)$ it follows that γ belongs to $\mathbf{R}(S) \cup \mathbf{R}_{\varepsilon}(S)$. Thus γ $\Rightarrow$ \$. If γ $\Rightarrow$ \$ then from the definition of EP relations we have $\gamma \in \mathbf{R}(S) \cup \mathbf{R}_{\varepsilon}(S)$. Thus there exists a derivation $S' ==> S ==>^{*}_{rm} \vartheta X ==>^{*}_{rm} \vartheta\alpha$, where $SUFF_1(\vartheta\alpha) = \gamma$. This implies the item $[X \rightarrow \alpha . \mid \varepsilon]$ belongs to $\Lambda(\gamma)$.

6.3.4 The item $[X \rightarrow \alpha . \mid v] \in \Lambda(\gamma)$, $\gamma \neq \varepsilon, v \neq \varepsilon$ if and only if γ $\Rightarrow$ v.

Proof (only if). Suppose $[X \rightarrow \alpha . \mid v] \in \Lambda(\gamma)$, where $\gamma \neq \varepsilon$ and $v \neq \varepsilon$. Then either $\gamma = SUFF_1(\alpha)$, or $\alpha = \varepsilon$ and there is an item $I = [A \rightarrow \vartheta\gamma . B\lambda \mid \omega]$ in $\Lambda(\gamma)$ such that $[X \rightarrow . \mid v] \in$ closure (I). We consider several subcases.

SUBCASE 6.3.4.a $\gamma = SUFF_1(\alpha)$.

In this case there must exist a production $A \rightarrow \delta B \vartheta C \beta \in P$, $B \in N$, $\vartheta \in NULL(G)^*$ and $C \in V_G$ such that $X \in \{B\} \cup \mathbf{R}(B) \cup \mathbf{R}_\varepsilon(B)$ and $v \subset \{C\} \cup \mathbf{L}(C) \cup \mathbf{L}_\varepsilon(C)$. It follows from Definition 3.2 that $\gamma \Leftrightarrow v$.

SUBCASE 6.3.4.b $\alpha = \varepsilon$.

Here we have $v \in \mathbf{L}_\varepsilon(B)$ or $v \in FIRST_1(\lambda\omega)$ and $B \in NULL(G)$. If $v \in \mathbf{L}_\varepsilon(B)$, then $\gamma \stackrel{+}{=} B$ implies at once that $\gamma \Leftrightarrow v$. If $v \subset FIRST_1(\lambda\omega)$ and $B \in Null(G)$, then either $v = \omega$ and $\lambda \in Null(G)^*$, or $\lambda = \vartheta C \beta$ where $\vartheta \in Null(G)^*$ and $v \in \{C\} \cup \mathbf{L}(C) \cup \mathbf{L}_\varepsilon(C)$. In the latter case we have $\gamma \stackrel{\varepsilon}{=} C$ and it follows directly from Definition 3.2 that $\gamma \Leftrightarrow v$. Finally, if $v = \omega$ and $B\lambda \in Null(G)^+$, then v is a valid right context of $A \rightarrow \delta \gamma B \lambda$ and there must exist a production $D \rightarrow x C_1 \vartheta C_2 y \in P$ such that $A \in \{C_1\} \cup \mathbf{R}(C_1) \cup \mathbf{R}_\varepsilon(C_1)$, $\vartheta \in Null(G)^*$ and $v \in \{C_2\} \cup \mathbf{L}(C_2) \cup \mathbf{L}_\varepsilon(C_2)$. Thus we have $\gamma \in \mathbf{R}_\varepsilon(C_1)$, $C_1 \stackrel{*}{=} C_2$ and $v \in \{C_2\} \cup \mathbf{L}(C_2) \cup \mathbf{L}_\varepsilon(C_2)$. It follows that $\gamma \Leftrightarrow v$.

Proof (if). Suppose $\gamma \Leftrightarrow v$, $\gamma \neq \varepsilon, v \neq \varepsilon$. Several cases arise.

SUBCASE 6.3.4.c $\gamma \stackrel{+}{=} Z$, $Z \in N$, and $v \in \mathbf{L}_\varepsilon(Z)$.

In this case $A \rightarrow \alpha \gamma Z \beta \in P$ and $I = [A \rightarrow \alpha\gamma . Z\beta \mid u] \in \Lambda(\gamma)$ for some $u \in \Sigma \cup \{\varepsilon\}$. By argument similar to the one given in (2) above there must be an item $[x \rightarrow . \mid v]$ in the closure(I) and thus in $\Lambda(\gamma)$.

SUBCASE 6.3.4.d $\gamma \stackrel{\varepsilon}{=} v$.

In this case there is $A \rightarrow \alpha \gamma Z_1 Z_2 \cdots Z_n v \beta \in P$ such that $Z_i \in NULL(G)$, $1 \leq i \leq n$. But then $I = [A \rightarrow \alpha\gamma . Z_1 Z_2 \cdots Z_n v \delta \mid u] \in \Lambda(\gamma)$ for some $u \in \Sigma \cup \{\varepsilon\}$.

Since $Z_1 \in$ Null(G), $Z_1 ==>^{+} \varepsilon$ and it follows that there is an item [X→. | t]∈ closure(I) for each $t \in FIRST_1(Z_2 \cdots Z_n v \delta u)$. Since $Z_i \in$ Null(G), it follows that [X→. | v]∈ closure(I)⊆$\Lambda(\gamma)$.

SUBCASE 6.3.4.e $\gamma \stackrel{\varepsilon}{=} B, \gamma \in \mathbf{L}(B) \cup \mathbf{L}_{\varepsilon}(B)$.

The argument is similar to the one given in the previous subcase. We observe that there is I=[A→$\alpha\gamma . Z_1 Z_2 \cdots Z_n B \delta \mid u$] $\in \Lambda(\gamma)$ such that $v \in FIRST_1(Z_2 \cdots Z_n B \delta u)$. The rest is obvious.

SUBCASE 6.3.4.f Y$\stackrel{*}{=} v, \gamma \in \mathbf{R}(Y) \cup \mathbf{R}_{\varepsilon}(Y)$.

In this case there is a production A→$\alpha Y \delta v \beta \in$ P such that $\delta ==>^{*}_{rm} \varepsilon$. Thus we have $S ==>^{*}_{rm} \vartheta A \omega ==> \vartheta \alpha Y \delta v \beta \omega ==>^{*}_{rm} \vartheta \alpha Y \delta v \omega' ==>^{*}_{rm} \vartheta \alpha Y v \omega'$. Now if $\gamma \in$ **R**(Y), then $\vartheta \alpha Y v \omega' ==>^{*}_{rm} \vartheta \alpha \lambda Z v \omega' ==>_{rm} \vartheta \lambda \Psi \gamma v \omega'$ and [Z→$\Psi\gamma . \mid v$]$\in \Lambda(\gamma)$. If $\gamma \in \mathbf{R}_{\varepsilon}$(Y), then we have $\vartheta \alpha Y v \omega' ==>^{*}_{rm} \vartheta \alpha \lambda Z v \omega' ==>_{rm} \vartheta \alpha \lambda \Psi B \beta v \omega'$, where $\gamma \subset \{B\} \cup \mathbf{R}(B)$ and $\beta ==>^{+}_{rm} \varepsilon$, $\beta \in N^{+}$. Thus $S ==>^{*}_{rm} \vartheta \alpha \lambda \Psi B \beta v \omega' ==>^{*}_{rm} \vartheta \alpha \lambda \Psi B v \omega'$ and by the argument given for $\gamma \in$ **R**(Y) it follows there is [C→$\Omega\gamma . \mid v$]$\in \Lambda(\gamma)$.

SUBCASE 6.3.4.g $Y_1 \stackrel{*}{=} Y_2$, Y_1, $Y_2 \in$ N, $\gamma \in \mathbf{R}(Y_1) \cup \mathbf{R}_{\varepsilon}(Y_1)$, $v \in \mathbf{L}(Y_2) \cup \mathbf{L}_{\varepsilon}(Y_2)$.

From $Y_1 \stackrel{*}{=} Y_2$, it follows there is a production A→$\alpha Y_1 \delta Y_2 \beta$ such that $\delta ==>^{*}_{rm} \varepsilon$. Thus we have a rightmost derivation in G of the form, S$==>^{*}_{rm} \vartheta A \omega ==>_{rm} \vartheta \alpha Y_1 \delta Y_2 \beta \omega ==>^{*}_{rm} \vartheta \alpha Y_1 \delta Y_2 \omega'$, where $\omega' \in \Sigma^{*}$. From $v \in \mathbf{L}(Y_2) \cup \mathbf{L}_{\varepsilon}(Y_2)$ and an argument styled after that given in (2) above we can find a rightmost derivation, π, such that $\vartheta \alpha Y_1 \delta Y_2 \omega' ==>^{\pi}_{rm} \vartheta \alpha Y_1 \delta v x ==>^{*}_{rm} \vartheta \alpha Y_1 v x$. By applying an argument similar to that given in (6.3.4.f) we obtain the desired result.

6.3.5 The item $[X\rightarrow .B\beta \mid v]\in\Lambda(\gamma)$, $\gamma\neq\varepsilon$ iff $\gamma \lhd B$.

Proof. If an item $[X\rightarrow .B\beta \mid v]\in\Lambda(\gamma)$, $\gamma\neq\varepsilon$, there must exist an item $I=[C\rightarrow\vartheta\gamma.D\lambda \mid \omega]$ in $\Lambda(\gamma)$ such that the item $[X\rightarrow .B\beta \mid v]$ belongs to closure(I). This implies $B\in\mathbf{L}(D)$ and since $\gamma \stackrel{+}{=} D$ from the definition of EP relations we have $\gamma \lhd B$. IF $\gamma \lhd B$, there must exist a production $C\rightarrow\vartheta\gamma.D\lambda$ and $X\subset N$ such that $X\in\mathbf{L}(D)\cup\{D\}$ and $X\rightarrow B\beta\in P$. $C\rightarrow\vartheta\gamma D\lambda\in P$ implies that the item $I=[C\rightarrow\vartheta\gamma.D\lambda \mid \omega]$ belongs to $\Lambda(\gamma)$. Since $X\rightarrow B\beta\in P$, the item $[X\rightarrow .B\beta \mid v]$ belongs to closure(I) $\subseteq$ $\Lambda(\gamma)$.

6.3.6 The item $[X\rightarrow .B\beta \mid v]\in\Lambda(\varepsilon)$, if and only if $\$ \lhd B$.

Proof. $[X\rightarrow .B\beta \mid v]\in\Lambda(\varepsilon)$ implies $X \in\{S\}\cup\mathbf{L}(S)$. Thus $B\in\mathbf{L}(S)$ and $\$ \lhd B$. The converse is equally straightforward.

6.3.7 The item $[X\rightarrow .\mid \varepsilon]\in\Lambda(\varepsilon)$, if and only if $(\$,\$)\in\rho(X)$.

Proof. If $[X\rightarrow .\mid \varepsilon]\in\Lambda(\varepsilon)$ then from (1) above we have $\$ \Rightarrow \$$. Also $X\neq S'$, $X\in L(S)$ and $X\in\mathbf{R}(S)$. Therefore $X \Rightarrow \$$ and $\$ \lhd X$. Thus $(\$,\$)\in\rho(X)$. If $(\$,\$)\in\rho(X)$, from definition of ρ we have $\$ \Rightarrow \$$ and from (1) above we have $[X\rightarrow .\mid \varepsilon]\in\Lambda(\varepsilon)$.

6.3.8 The item $[X \rightarrow .\mid v]\subset\Lambda(\varepsilon)$, $v \neq\varepsilon$, if and only if $(\$,v)\subset\rho(X)$.

Proof. If $[X\rightarrow .\mid v]\in\Lambda(\varepsilon)$ then from (2) above we have $\$ \Rightarrow v$. Also the item $[X\rightarrow .\mid v]$ in $\Lambda(\varepsilon)$ implies that $X\in L(S)$. Therefore $\$ \lhd X$ and $\$ \Rightarrow v$. Thus $(\$,v)\in\rho(X)$. If $(\$,v)\in\rho(X)$ from definition of ρ we have $\$ \Rightarrow v$ and from (2) above we have item $[X\rightarrow .\mid v]\in\Lambda(\varepsilon)$.

6.3.9 The item $[X\rightarrow .\mid \varepsilon]\subset\Lambda(\gamma)$, $\gamma\neq\varepsilon$ if and only if $(\gamma,\$)\in\rho(X)$.

Proof. If $[X\rightarrow .\mid \varepsilon]\in\Lambda(\gamma)$, then from Lemma 6.3.3 we have $\gamma \Rightarrow \$$. Also the item $[X\rightarrow .\mid \varepsilon]\in\Lambda(\gamma)$ implies that there exists an item $I=[C\rightarrow\vartheta\gamma.A\lambda \mid \varepsilon]$ in $\Lambda(\gamma)$ such that

$[X\rightarrow .|\varepsilon]\in$ closure(I) and $A\lambda ==>^{*}_{rm}\varepsilon$. This implies $X\in L(A)\cup\{A\}$ therefore $(\gamma,X)\in \stackrel{\pm}{=}\cup <\!\!\cdot$. Also the above item implies that $X\in R(S)\cup R_{\varepsilon}(S)$ therefore $(X,\$)\in \cdot\!\!>$. Thus $(\gamma,\$)\in\rho(X)$. If $(\gamma,\$)\in\rho(X)$, from the definition of ρ we have $(X,\$)\in \cdot\!\!>$ and from Lemma 6.3.3 we have $[X\rightarrow .|\varepsilon]\in\Lambda(\gamma)$.

6.3.10 The item $[X\rightarrow .|v\,]\in\Lambda(\gamma)$, $\gamma\neq\varepsilon$, $v\neq\varepsilon$, if and only if $(\gamma,v\,)\in\rho(X)$.

Proof. If item $[X\rightarrow .|v\,]\in\Lambda(\gamma)$, then from Lemma 6.3.4 we have $\gamma \cdot\!\!> v$. Also the item $[X\rightarrow .|v\,]\in\Lambda(\gamma)$ implies there is an item $I=[A\rightarrow \vartheta\gamma.C\beta\,|\,\lambda]$ in G such that $[X\rightarrow .\,|\,v\,]\in$ closure (I). But this implies $X\in L(C)\cup\{C\}$ and it follows that $(\gamma,X)\in \stackrel{\pm}{=}\cup <\!\!\cdot$. Furthermore, $[X\rightarrow .\,|\,v\,]\in$ closure(I) implies $(X,v\,)\in(\stackrel{\pm}{=}\cup <\!\!\cdot\cup \cdot\!\!>)$. Thus $(\gamma,v\,)\in\rho(X)$. If $(\gamma,v\,)\in\rho(X)$, from the definition of ρ we have $(X,v\,)\in \cdot\!\!>$ and from Lemma 6.3.4 we have $[X\rightarrow .|v\,]\in\Lambda(\gamma)$.

We are now prepared to prove that every EWP grammar is ESR(1,1).

THEOREM 6.4 Every EWP grammar is ESR(1,1).

Proof. This result will be established by contradiction showing that if any one of conditions (ii-a), (ii-b) or (ii-c) of Definition 6.1 is violated for some reduced context-free grammar, G, then G is not EWP.

CASE 1 (condition (ii-a) is violated)

Assume therefore that $[Z\rightarrow\alpha S.\beta\,|\,v\,]\in\Lambda(S)$, where $Z\neq S'$ and each of the following conditions holds:

(1.1) $\varepsilon\in FIRST_1(\beta v\,)$,

(1.2) $\alpha=\varepsilon$,

(1.3) $[Z\rightarrow .S\beta\,|\,v\,]\in\Lambda(\varepsilon)$.

Together these imply $v=\varepsilon$ and $S ==>^{*}_{rm} Z\delta ==>^{*}_{rm} Z ==>^{*}_{rm} S\beta ==>^{*}_{rm} S$. Thus G contains a cycle and cannot be EWP.

CASE 2 (condition (ii-b) is violated)

Assume the following for some γ, $\gamma' \in V_G^{*1}$.

(2.1) $[A \rightarrow \alpha.\beta \mid u] \in \Lambda(\gamma)$,

(2.2) $[B \rightarrow \delta\alpha. \mid v] \in \Lambda(\gamma)$,

(2.3) $v \in EFF_1(\beta u)$,

(2.4) $[A \rightarrow .\alpha\beta \mid u] \in \Lambda(\gamma')$,

(2.5) $[B \rightarrow \delta.\alpha \mid v] \in \Lambda(\gamma')$.

We consider two different subcases.

SUBCASE 2.1 ($\gamma=\varepsilon$).

In this case $\delta=\alpha=\gamma'=\varepsilon$. Since $v \neq \varepsilon$, then by condition (2.2) and Lemma 6.3.2, $\$ \Leftrightarrow v$. Furthermore, by Lemma 6.3.8, $(\$,v) \in \rho(B)$. Now if $\beta=\varepsilon$, then conditions (2.1) and (2.3) imply $v=u$ and by Lemma 6.3.8, $(\$,v) \in \rho(A)$. Thus $\rho(A) \cap \rho(B) \neq \phi$ for A≠B and G cannot be EWP. Alternatively, if $\beta \neq \varepsilon$, then condition (2.3) and Lemma 6.3.6 imply $\$ \lessdot v$. But since $\$ \Leftrightarrow v$, $\lessdot$ and $\Leftrightarrow$ are not disjoint and G cannot be EWP.

SUBCASE 2.2 ($\gamma \neq \varepsilon$). Suppose $\alpha=\varepsilon$. Then $\gamma'=\gamma$. Since $v \neq \varepsilon$, (2.2) and Lemma 6.3.4 imply $\gamma \Leftrightarrow v$. If $\beta=\varepsilon$, then there must be an item, $I=[Z \rightarrow \vartheta\gamma.C\delta \mid x] \in \Lambda(\gamma)$, such that $A \in \{C\} \cup L(C)$. Thus $\gamma \stackrel{+}{=} A$ or $\gamma \lessdot A$. If $\beta=\varepsilon$ and $\delta=\varepsilon$, then Lemma 6.3.10, and conclusions (2.1) and (2.2) imply that $\rho(A) \cap \rho(B) \neq \phi$ and G cannot be EWP. If $\beta=\varepsilon$ and $\delta \neq \varepsilon$, then $\gamma=SUFF_1(\delta)$ and $(\gamma,A) \in (\stackrel{+}{=} \cup \lessdot)$ violates Definition 3.9 and G cannot be EWP. If $\beta \neq \varepsilon$, then by Lemma 6.3.5, $\gamma \lessdot PREF_1(\beta)$. Now either

$v = PREF_1(\beta)$ or $v \in L(PREF_1(\beta))$. In either case it follows that $\gamma \lessdot v$. But since $\gamma \gtrdot v$ as well, we have a contradiction to Definition 3.9 and G is not EWP.

Now suppose $\alpha \neq \varepsilon$. In this case $\gamma = SUFF_1(\alpha)$ and by Lemma 6.3.4, and conclusion (2.2) implies $\gamma \gtrdot v$. Also if, $\beta \neq \varepsilon$, then (1) and (3) imply $\gamma \stackrel{\pm}{=} v$ or $\gamma \lessdot v$; in either case $(\stackrel{\pm}{=} \cup \lessdot) \cap \gtrdot \neq \phi$ and G is not EWP. If $\beta = \varepsilon$, then $u = v$. From (4) it follows $\gamma' \stackrel{\pm}{=} A$ or $\gamma' \lessdot A$. Thus, if $\delta \neq \varepsilon$, then $\gamma' = SUFF_1(\delta)$ and we obtain a contradiction to Definition 3.9. In the case $\delta = \varepsilon$, $A \neq B$ implies a violation to almost unique invertibility and G cannot be EWP.

CASE 3 (condition (ii-c) is violated)

Here we assume the following hold for some $\gamma, \gamma' \in V_G^{*1}$.

(3.1) $[A \rightarrow \delta\alpha.\beta \mid u] \in \Lambda(\gamma)$,

(3.2) $[B \rightarrow \alpha. \mid v] \in \Lambda(\gamma)$,

(3.3) $v \in EFF_1(\beta u)$,

(3.4) $[A \rightarrow \delta.\alpha\beta \mid u] \in \Lambda(\gamma')$,

(3.5) $[B \rightarrow .\alpha \mid v] \in \Lambda(\gamma')$.

The arguments here are very similar to those presented for case 2. We observe that if both δ and β are null, then either $\rho(A) \cap \rho(B) \neq \phi$, or G is not AUI. If $\gamma = \varepsilon$, then $\delta = \alpha = \varepsilon$ and we can show $(\$, v) \in (\lessdot \cap \gtrdot)$. If $\gamma \neq \varepsilon$, then $\alpha = \varepsilon$ implies $(\gamma, v) \in (\lessdot \cap \gtrdot)$. For $\gamma \neq \varepsilon$ and $\alpha \neq \varepsilon$ $\beta \neq \varepsilon$ implies $(\gamma, v) \in (\lessdot \cap \gtrdot)$. In case $\beta = \varepsilon$ (and $\delta = \varepsilon$) we observe $\gamma = SUFF_1(\delta)$ and that $\gamma \stackrel{\pm}{=} B$ or $\gamma \lessdot B$ giving a violation to Definition 3.9. Thus in every case contradictions arise to Definition 3.6 and G cannot be EWP.

Three important results follow as corollaries of theorem 6.5.

COROLLARY 1. The class of ESR(1,1) languages is co-extensive with the class of deterministic context-free languages.

Proof. We have shown in Chapter 4 that every deterministic context-free language is generated by some EWP grammar. The inclusion of EWP grammars in ESR(1,1) grammars implies that every deterministic context-free language is defined by some ESR(1,1) grammar. Moreover, Theorem 6.2 implies that ESR(1,1) languages are deterministic and the result follows.

COROLLARY 2 The class of EWP languages is co-extensive with the class of deterministic context-free languages.

Proof. The inclusion of EWP grammars in the class of ESR(1,1) grammars implies that EWP grammars define only deterministic context-free languages. Additionally, from Theorem 4.7 every deterministic context-free language is defined by some EWP grammar and the result follows.

COROLLARY 3. The hierarchy of ESP languages exhausts the class of deterministic context-free languages.

Proof. Obvious.

Meaningful comparison of EWP grammars with simple mixed strategy and left context precedence grammars is not possible, since these grammars do not have ε-rules. Therefore, we have generalized the classes of simple mixed strategy and left context precedence grammars to allow for ε-rules. The resulting classes of grammars are then compared with ESP grammars.

DEFINITION 6.5 A reduced context-free grammar, G=(N,Σ,P,S), is said to be an *Epsilon Mixed Precedence* (EMP) grammar if

(1) G is cycle-free;

(2) Epsilon precedence relations for G are pairwise disjoint;

(3) for all Z', $Z'' \in N_\varepsilon(G)$, $Z' \neq Z''$, $\rho(Z') \cap \rho(Z'')=\phi$ where for all $Z \in N_\varepsilon(G)$, $\rho(Z)$ is defined as follows:

$$\rho(Z)=\{(X,t) \mid (X,Z) \in (\stackrel{\pm}{=} \cup \lessdot), (X,t) \in \doteq\!\!>, (Z,t) \in (\stackrel{\pm}{=} \cup \lessdot \cup \doteq\!\!>)\};$$

(4) if A $\rightarrow \alpha$X and B$\rightarrow \varepsilon$ are two distinct productions in P, then (X,B) $\notin \stackrel{\pm}{=} \cup \lessdot$;

(5) for all $Z \in N_\varepsilon(G)$, (S,\$) $\notin \rho(Z)$;

(6) if A$\rightarrow \alpha$ and B$\rightarrow \alpha$, $\alpha \neq \varepsilon$ are distinct productions in P, then $\iota(A) \cap \iota(B)=\phi$, where for all $Y \in N$, $\iota(Y)=\{X \mid (X,Y) \in (\stackrel{\pm}{=} \cup \lessdot)\}$

Definition 6.5 is similar to the definition of simple mixed strategy precedence grammars [5] except ε-rules are allowed and instead of Wirth-Weber precedence relations EP relations are used. Conditions (4) and (5) are added to avoid possible conflicts that may arise as the result of allowing for ε-rules. The following corollary follows immediately from Definition 6.5.

COROLLARY The class of ESP grammars is properly included in the class of EMP grammars.

Proof. To show that ESP grammars are included in the class of EMP grammars, we only observe that conditions (1)-(5) are identical to conditions (2)-(6) of the definition of ESP grammars and that condition (6) is always satisfied by Almost Uniquely Invertible grammars. For proper inclusion, we just observe that EMP grammars need not be Almost Uniquely Invertible.

We now turn our attention to the class of left context precedence grammars. Our objective here is to generalize left context precedence grammars to allow for ε-rules. We first give a revised definition for the set of goal variables which are used to define left context precedence grammars.

DEFINITION 6.6 Let G=(N,Σ,P,S) be a precedence grammar, for which the EP relations are pairwise disjoint. Assume that $\gamma \in V^+$ is a viable prefix of G with prefix representation $\psi_0 \cdots \psi_n$ of $\$\gamma$.

(1) The set GL(ψ_0)=GL(\$)={S} is called the set of goal variables of ψ_0.

(2) For $i=1, \cdots, n$ the set

GL($\psi_0 \cdots \psi_i$)={$A \in N$ | there exists a derivation

$$\$S\$ ==>_{rm}^{*} \psi_0\psi_1 \cdots \psi_{i-1} Ax\$ ==>_{rm}^{*}$$

$$\psi_0\psi_1 \cdots \psi_i \lambda_i x\$, x \in \Sigma^*, \psi_i\lambda_i \in V^+\}$$

$$\cup\{(Z, PREF_1(x\$)) \mid Z \in N \text{ and there exists a derivation}$$

$$\$S\$ ==>_{rm}^{*} \psi_0\psi_1 \cdots \psi_i Zx\$ ==>_{rm} \psi_0\psi_1 \cdots \psi_i x\$, \ x \in \Sigma^*\}$$

is called the set of goal variables for $\$\gamma$.

DEFINITION 6.7 Let G=(N,Σ,P,S) be a reduced context-free grammar. G is said to be Epsilon Left Context Precedence (ELCP) if

(1) G is cycle-free;

(2) EP relations are pairwise disjoint for G;

(3) for all $z \in \Sigma^*$, for all viable prefixes of G, γ, with prefix representation $\psi_0 \cdots \psi_n$ of $\$\gamma$, the function VP, defined below is uniquely defined for G.

$$VP(\$\gamma,z\$)=\begin{cases}(\gamma a,z'\$) & \text{if } z=az',\ SUFF_1(\gamma)\stackrel{\pm}{=} a \text{ and } \textit{there exists}\\ & A\in GL(\psi_0\cdots\psi_n) \textit{ such that}\\ & A\rightarrow\psi_n a\lambda_n\in P, \text{for } \textit{some } \lambda_n\in V^*\\ (\gamma a,z'\$) & \text{if } z=az',\ SUFF'_1(\gamma)\leftarrow a \text{ and } \textit{there exists}\\ & B\in GL(\psi_0\cdots\psi_n) \textit{ such that}\\ & B\rightarrow\psi_n C\lambda_n\in P \text{ and } C\Longrightarrow^*_{rm} a\lambda\in P,\\ & \text{for } \textit{some } \lambda, \text{ and } \lambda_n \textit{ in } V^*\\ (\psi_0\cdots\psi_{n-1}B,z\$) & \text{if } SUFF_1(\gamma)\Leftrightarrow PREF_1(z) \text{ and } \textit{there exists}\\ & B\in GL(\psi_0\cdots\psi_n) \textit{ such that } B\rightarrow\psi_n\in P\\ (\psi_0\cdots\psi_n Z,z\$) & \text{if } SUFF'_1(\gamma)\Leftrightarrow PREF'_1(z\$) \text{ and } \textit{there exists}\\ & (Z,PREF_1(x\$))\in GL(\psi_0\cdots\psi_n)\\ UNDEFINED & OTHERWISE\end{cases}$$

The above definition is identical to the definition of LCP grammars [35] except that instead of Wirth-Weber precedence relations, EP relations are used and in case of conflicts involving ε-rules in addition to the contents of the stack, one symbol of the lookahead is examined.

THEOREM 6.8 The class of EWP grammars is properly included in the class of ELCP grammars.

Proof. Let G be an ESP grammar. Clearly, G satisfies conditions (1) and (2) of definition 6.7. Let γ be a viable prefix of G with prefix representation of $\psi_0\cdots\psi_n$ of $\$\gamma$ and z be a symbol in $\Sigma\cup\{\varepsilon\}$. Assume for the sake of contradiction that VP($\$\gamma,z\$$) is multiply defined. Obviously, since G is Almost Uniquely Invertible and have unique EP relations, there must exist B and Z, B$\in$N, Z$\in N_\varepsilon$(G), such that B and (Z, $SUFF_1(x\$)$) belong to GL($\psi_0\cdots\psi_n$) where B$\rightarrow\psi_n$, $\psi_n\neq\varepsilon$ is

a production in P. Suppose $\psi_n = \psi'_n X$. Then, $SUFF_1(\psi_n)$ = $SUFF_1(\gamma)$ = X and there exists a derivation

$$\$S\$ ==>^*_{rm} \psi_0\psi_1 \cdots \psi_n Zx\$ = \psi_0\psi_1 \cdots \psi'_n XZx\$ ==>^*_{rm}$$
$$\psi_0\psi_1 \cdots \psi'_n Xx\$,\ x \in \Sigma^*$$

in G. Hence, (X,$PREF_1(z\$)$)$\in\rho(Z)$, and we must have that (X,Z)$\in(\stackrel{\pm}{=} \cup \lessdot)$. But, P contains productions B$\rightarrow\psi'_n$ 'X, and Z$\rightarrow\varepsilon$, and (X,Z)$\in(\stackrel{\pm}{=} \cup \lessdot)$. Thus, G is not ESP and we have a contradiction to the assumption that G is ESP.

We now compare strict deterministic and ESP languages, and show that for all n, and i, the class of strict deterministic languages of degree n is incommensurate with the class of ESP_i languages.

LEMMA 6.9 For all n, and i, $i\geq 0, n\geq 1$, the class of strict deterministic languages of degree n is incommensurate with the class of ESP_i languages.

Proof. Recall from Definition 2.16 that strict deterministic languages of degree n are those that may be recognized by a DPDA having n states with empty stack in a final state. For each n, and i, $i\geq 0$, $n\geq 1$, consider the language

$$L_{n,i} = \{x^m y^k x^m y^k \mid m\geq 1,\ 1\leq k\leq n\} \cup \{\bigcup_{j=1}^{i+1} a_j 0^n 1^{j\times n} \mid n\geq 1\}$$

Using the same argument presented in Harrison[22] showing that the language $L_n = \{x^m y^k x^m y^k \mid m\geq 1,\ 1\leq k\leq n\}$ is strict deterministic of degree at least n, we may show that $L_{n,i}$ is also strict deterministic of degree at least n. Similarly, using the argument presented in chapter 5, we can show that $L_{n,i}$ is EWP_j where j is at least equal to i. We now show that $L_{n,i}$ is indeed ESP_i and strict deterministic of degree n. First, we present an ESP_i grammar defining $L_{n,i}$.

$G_{n,i}$:

S $\rightarrow S_1$

S $\rightarrow S_2$

$S_1 \rightarrow a_j X_j$ $\quad 1 \leq j \leq i+1$

$X_j \rightarrow A_j\ X_j\ C_j 1$ $\quad 2 \leq j \leq i+1$

$X_j \rightarrow A_j\ C_j 1$ $\quad 2 \leq j \leq i+1$

$A_j \rightarrow Z_j\ 0$ $\quad 2 \leq j \leq i+1$

$C_j \rightarrow C_{j-1} 1$ $\quad 3 \leq j \leq i+1$

$C_2 \rightarrow 1$

$Z_j \rightarrow \varepsilon$ $\quad 2 \leq j \leq i+1$

$X_1 \rightarrow A_1 X_1 1$

$X_1 \rightarrow A_1 1$

$A_1 \rightarrow 0$

$S_2 \rightarrow DE_k x Y_k$ $\quad 1 \leq k \leq n$

$E_k \rightarrow DE_k x$ $\quad 1 \leq k \leq n$

$E_k \rightarrow Y_k$ $\quad 1 \leq k \leq n$

$Y_k \rightarrow y^k$

D $\rightarrow x$

It can be easily shown that the above grammar is ESP_i and generates exactly $L_{n,i}$. Hence $L_{n,i}$ is ESP_i. Now, we give a dpda with n states recognizing exactly the sentences in $L_{n,i}$.

$\delta(q_0,x,Z_0)=(q_0,AB)$

$\delta(q_0,x,B)=(q_0,CB)$

$\delta(q_0,y,B)=(q_0,D)$

$\delta(q_k,y,D)=(q_{k+1},D)$ $\quad 0\leq k<n-1$

$\delta(q_k,x,D)=(q_k,\varepsilon)$ $\quad 0\leq k\leq n-1$

$\delta(q_k,y,A)=(q_{k-1},\varepsilon)$ $\quad 0<k\leq n-1$

$\delta(q_0,y,A)=(q_0,\varepsilon)$

$\delta(q_0,a_j,Z_0)=(q_0,[a_j])$ $\quad 1\leq j\leq i+1$

$\delta(q_0,0,[a_j])=(q_0,[a_j,0^j])$ $\quad 1\leq j\leq i+1$

$\delta(q_0,0,[a_j,0^j])=(q_0,[0]^j[a_j,0^j])$ $\quad 1\leq k\leq i+1$

$\delta(q_0,1,[a_j,0^l])=(q_0,[a_j,0^{l-1}])$ $\quad 1\leq j\leq i+1,\ \ l>0$

$\delta(q_0,1,[a_j,0])=(q_0,\varepsilon)$ $\quad 1\leq j\leq i+1$

$\delta(q_0,1,[0])=(q_0,\varepsilon)$

Again, it can be easily verified that the n-state dpda given above, recognizes $L_{n,i}$ in the final state q_0, with empty stack. Thus, we have shown the following

(1) For a fixed n, the class of strict deterministic languages of degree n includes at least one ESP_i language, for all $i\geq 0$.

(2) For a fixed i, the class of ESP_i languages contain at least of strict deterministic language of degree n, for all $n\geq 1$.

Conclusions (1) and (2) above establish our desired result.

COROLLARY For all n, and i, $i\geq 0, n\geq 1$, the class of strict deterministic grammars of degree n is incommensurate with the class of ESP_i grammars.

Proof. Obvious.

We conclude our comparisons by comparing ESP and LL languages.

LEMMA 6.10 For all k, and i, $i \geq 0$, $k \geq 1$, the class of LL(k) languages is incommensurate with the class of ESP_i languages.

Proof. It is known that the classes of LL(1) and ESP_0 (simple precedence) languages are incommensurate. Hence, we assume that $i \geq 1, k > 1$. Consider the language

$$L_{k,i} = \{x^n \omega \mid n \geq 1 \text{ and } \omega \subset \{b, c, b^{k+1} d\}^n\} \cup \{\bigcup_{j=1}^{i+2} a_j 0^n 1^{j \times n} \mid n \geq 1\}$$

Using the arguments given in Aho and Ullman [5], showing the language $\{x^n \omega \mid n \geq 1 \text{ and } \omega \in \{b, c, b^{k+1}\}^n,\}$ is not LL(k), we could establish that $L_{k,i}$ is not LL(k). Similarly, we can show that $L_{k,i}$ is not ESP_i. To complete the proof, show that $L_{k,i}$ is LL($k+1$) and ESP_{i+1}. The following is a LL(k) grammar which generates $L_{k,i}$.

$S \rightarrow S_1$

$S \rightarrow S_2$

$S_1 \rightarrow xA$

$A \rightarrow S_1 B$

$A \rightarrow B$

$B \rightarrow bC$

$B \rightarrow c$

$C \rightarrow b^k d$

$C \rightarrow \varepsilon$

$S_2 \rightarrow a_j X_j$ $\quad 1 \leq j \leq i+2$

$X_j \rightarrow 0 Y_j$ $\quad 1 \leq j \leq i+2$

$Y_j \rightarrow X_j 1^j$ $\quad 1 \leq j \leq i+2$

$Y_j \rightarrow 1^j$ $\quad 1 \leq j \leq i+2$

Now consider the grammars G=(N,Σ,P,S) and $G'=(N',\Sigma',P',S)$ given below.

G_i

$S \rightarrow a_k X_k$	$1 \leq k \leq i+1$
$X_k \rightarrow D_k\ X_k\ E_k$	$1 \leq k \leq i+1$
$X_k \rightarrow D_k\ E_k$	$1 \leq k \leq i+1$
$D_k \rightarrow Z_k\ 0$	$1 < k \leq i+1$
$E_k \rightarrow E_{k-1}\ 1$	$1 < k \leq i+1$
$Z_k \rightarrow \varepsilon$	$1 < k \leq i+1$
$D_1 \rightarrow 0$	
$E_1 \rightarrow 1$	

G_k

S→$[xc]$

S→$[xb^{k+1}d]$

S→$[x^i, xb^{i+1}]$, $0 \le i \le k$

$[x^i, xb^j] \rightarrow [x][x^{i-1}, xb^j]$ $1 \le i \le j \le k+1$

$[\varepsilon, xb^j] \rightarrow [xb^j]$ $1 \le j \le k+1$

$[xb^{j+1}] \rightarrow [xb^j][b]$ $1 \le j \le k+1$

$[xb^{k+1}] \rightarrow [x, xb^{k+1}][b]$

$[xb] \rightarrow [x][b]$

$[xb] \rightarrow [x][xb^{k+1}d][b]$

$[xb] \rightarrow [x][xc][b]$

$[xb^{k+1}d] \rightarrow [xb^{k+1}][d]$

$[xc] \rightarrow [x][xb^{k+1}d][c]$

$[xc] \rightarrow [x][xc][c]$

$[xc] \rightarrow [x^i, xb^i][c]$ $1 \le i \le k+1$

$[xc] \rightarrow [x][c]$

$[x] \rightarrow x$

$[b] \rightarrow b$

$[c] \rightarrow c$

$[d] \rightarrow d$

The EP relations for G_k are shown in Table 8.

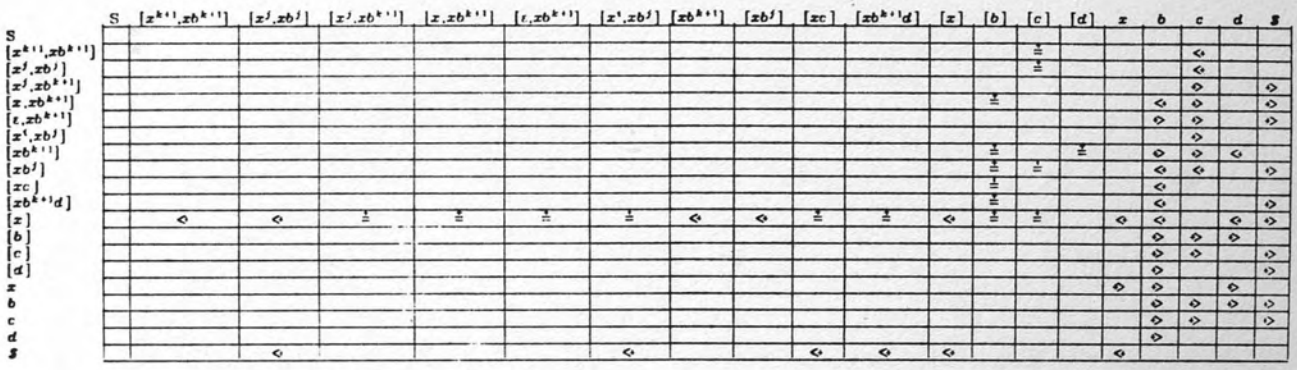


Table 8. The EP relations for G_k, $0 \leq i < j \leq k$.

Clearly, G_i is EWP_i and G_n is EWP_0. Let

$$G_{n,i}=(\{S\}\cup N\cup N',\Sigma\cup\Sigma',\{S'\rightarrow S\}\cup P\cup P',S').$$

Then, $L_{n,i}=L(G_{k,i})$. Therefore, for all k, and i, $i\geq 0$, $k\geq 1$, the class of LL(k) languages is incommensurate with the class of ESP_i languages.

The relationship between ESP languages and the other classes of languages is summarized in Figure 8.

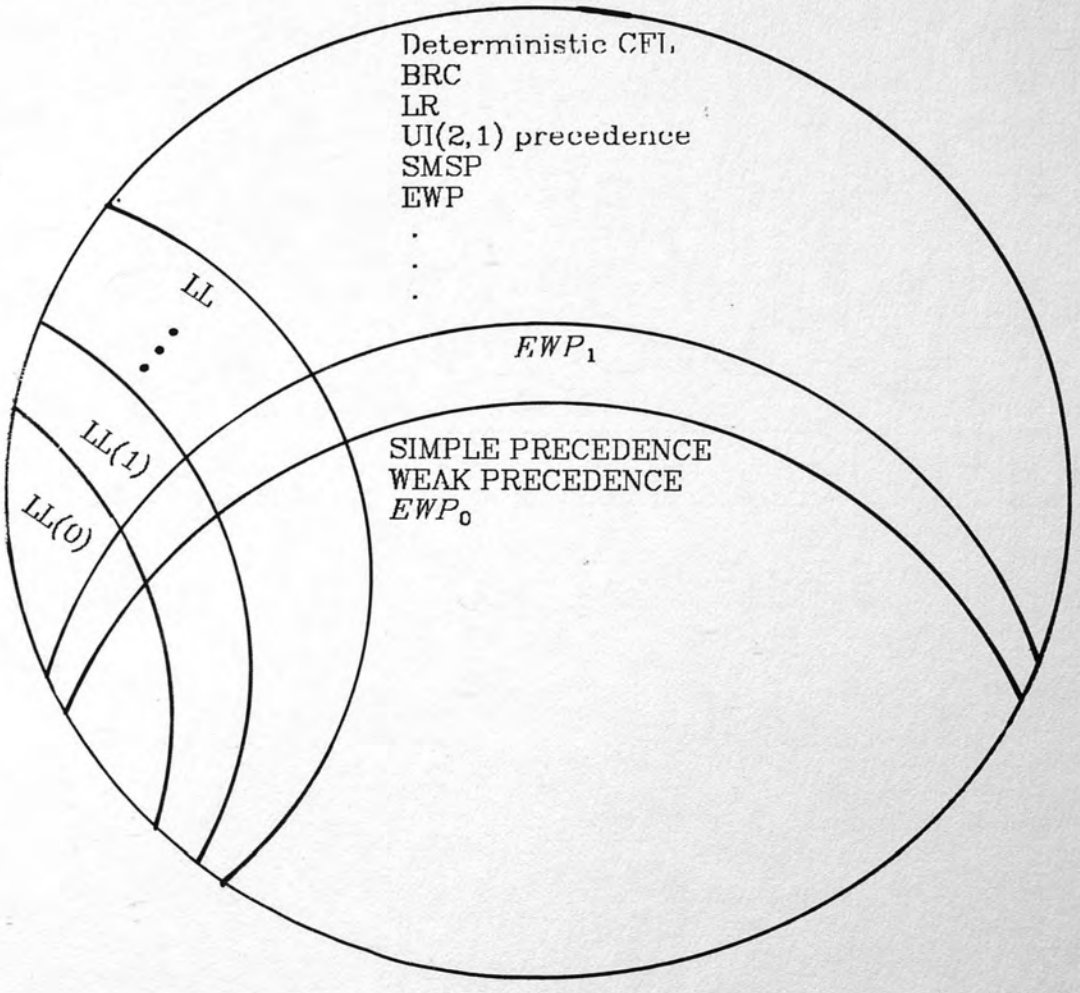


Figure 8. The hierarchy of deterministic context-free languages

CHAPTER 7

CONCLUSIONS

This chapter summarizes this research and suggests further research directions.

7.1 Summary

This research has developed the theory of epsilon precedence grammars and languages. We have generalized different classes of precedence grammars by allowing for ε-rules. This generalization is not trivial since we have shown that the class of epsilon simple precedence grammars describes all deterministic context-free languages; that is, they are equivalent in descriptive power to LR(1) grammars.

The existence of an infinite hierarchy among epsilon precedence languages was established. We demonstrated that for all $i \geq 0$, the class of ESP_i languages is properly included in the class of ESP_{i+1} languages. To establish the hierarchy of ESP languages, we have proven a new iteration theorem for the class of ESP_i languages. This is the only exhaustive hierarchy of deterministic context-free languages studied in the literature. We have also shown that the hierarchy of epsilon precedence languages is orthogonal to the hierarchies of strict deterministic and LL languages; that is, for each $i, j \geq 0$, $L_i - L_j \neq \phi$ and $L_j - L_i \neq \phi$, where L_i is the class of ESP_i languages and L_j is either the class of LL(j) languages or the class of strict deterministic languages of degree j.

Algorithms were developed to convert arbitrary LR(1) grammars directly to equivalent (2,1)EWP and EWP grammars. The EWP grammars produced by these algorithms are Viable Prefix EWP grammars. We demonstrated that EWP parsers for Viable Prefix EWP grammars have the viable prefix property. That is, they detect syntactic errors at the earliest possible time. This result is of particular importance since it shows for the first time that every deterministic context-free language has a (1,1) epsilon precedence parser with the viable prefix property.

Finally, we proved that the class of Viable Prefix EWP grammars is properly contained in the well-known classes of context-free grammars with ε-rules defining exactly the deterministic context-free languages, thereby characterizing the most restrictive set of known properties for a class of grammars defining every deterministic context-free language while yielding viable prefix parsers.

7.2 FUTURE DIRECTION OF RESEARCH

There are a number of ways in which our research may be extended. We conclude by enumerating some of them.

(1) The implementation of a parser-generating system based on epsilon precedence techniques is the first obvious extension of our research. The epsilon precedence parser-generating system must accept LR(1) grammars as input and produce a compact epsilon precedence parser as output. The effectiveness of different parser optimization schemes may be analyzed with this system.

(2) The algorithms developed in Chapter 4 may prove to be the useful in the development of incremental parsers. These algorithms encode the LR state information into the syntax tree, permitting the effective restoration of the parser to any configuration using only syntax tree.

(3) A number of important theoretical questions remains open. The equivalence problem for ESP_i grammars and deciding whether or not an ESP_i language is also ESP_j, for some $j<i$ were not addressed in our research. Also, the relationship of UI(1,k) precedence grammars [4] to ESP_i grammars is not established in our research.

(4) Although we established that the class of Viable Prefix EWP grammars defines exactly the deterministic context-free languages, a number of questions concerning these grammars remains unanswered. Among these are the equivalence of Viable Prefix EWP_i and Viable Prefix ESP_i grammars and determining whether or not every EWP_i language is defined by some Viable Prefix EWP_i grammar.

LIST OF REFERENCES

1. Aho, A.V. "Translator Writing Systems: Where Do They Stand," IEEE Computer 13 (August 1980): 9-14.

2. Aho, A.V.; Denning, P.J.; and Ullman, J.D. "Weak and Mixed Strategy Parsing," Journal of ACM 19 (Feburary 1972): 225-243.

3. Aho, A.V., and Johnson, S.C. "LR Parsing," Computing Surveys 6 (Feburary 1974): 99-124.

4. Aho, A.V., and Ullman, J.D. Principles of Compiler Design. Reading, Mass.: Addison-Wesley, 1977.

5. Aho, A.V., and Ullman, J.D. The Theory of Parsing, Translation and Compiling: vols. 1 and 2. Englewood Cliffs, N.J.: Prentice-Hall, 1972.

6. Aoe, J.; Yamamoto, Y.; and Shimada, R. "A Practical Method for Reducing Weak Precedence Parsers," IEEE Transactions on Software Engineering SE-9 (January 1983).

7. Backhouse, R.C. Syntax of Programming Languages - Theory and Practice. Englewood Cliffs, N.J.: Prentice-Hall, 1977.

8. Backus, J.W. "The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference," Proceedings of the International Conference on Information Processing. Paris: UNESCO (1959).

9. Bar-Hillel, Y.; Perles, M.; and Shamir, E. "On formal Properties of Simple Phrase Structure Grammars," Z. Phonetik Sprachwiss. Kommunicat. 14 (1961): 143-172.

10. Beaty, L. "Two Iteration Theorems for LL(k) Languages," Journal of Computer and System Sciences 12 (1980): 193-228.

11. Bell, J.R. "A New Method for Determining Linear Precedence Functions for Precedence Grammars" Communications of ACM12

12. Chomsky, N. "On Certain Formal Properties of Grammars," Infomation Control 2 (Feburary 1959): 137-167.

13. Crespi-Reghizzi, S.; Guida, G.; and Mandrioli, D. "Operator Precedence Grammars and the Noncounting Property," SIAM Journal Computing 10 (Janurary 1981): 174-191.

14. DeRemer, F.L., and Pennello, T.J. "Efficient Computation of LALR(1) Lookahead Sets," ACM Transactions on Programming Languages and Systems. 4 (April 1982): 615-649.

15. Fischer, M.J. "Some Properties of Precedence Languages," Proceedings of ACM Symposium on Theory of Computing (1969): 181-190.

16. Floyd, R.W. "Bounded Context Syntactic Analysis," Communications of ACM 7 (Feburary 1964): 62-67.

17. Floyd, R.W. "The Syntax of Programming Languages - A Survey," IEEE Transactions on Electronic Computers EC-13 (1964): 346-353.

18. Graham, S.L. "Extended Precedence Languages, Bounded Right Context Languages and Deterministic Languages," IEEE Conference Record of the 11th Annual Symposium on Switching and Automata Theory(1970):

19. Graham, S.L. and Rhodes, S.P. "Practical Syntactic Error Recovery," Communications of ACM 18 (November 1975): 639-650.

20. Gray, J.N. "Precedence Parsers for Programming Languages." Ph.D. Dissertation, University of California, Berkeley, California: 1969.

21. Gray, J.N. and Harrison, M.A. "Single Path Precedence Analysis," IEEE Conference Record of the 10th Annual Symposium on Switching and Automata Theory (1969): 106-117.

22. Harrison, M.A. Introduction to Formal Language Theory. Reading, Mass.: Addison-Wesley, 1978.

23. Harrison, M.A. and Havel, I.M. "Strict Deterministic Languages," Journal of Computer and Systems Sciences 7 (1973): 237-277.

24. Ichbiah, J.D. and Morse, S.P. "A Technique for Generating Almost Optimal Floyd-Evans Productions for Precedence Grammars," Communications of ACM 13 (August 1970): 501-508.

25. King, K.N. "Iteration Theorems for Families of Strict Deterministic Languages," Theoretical Computer Sciencs 10 (1980): 317-333.

26. Knuth, D.E. "On the Translation of Languages from Left to Right," Information Control 8 (June 1965): 607-639.

27. Koskimies, K. and Soisalon-Soininen, E. "On a Method for Optimizing LR Parsers," International Journal of Computer Mathematics 7 (1979): 287-296.

28. Krevner, Y. and Yehudai, A. "An Iteration Theorem for Simple Precedence Languages," Journal of ACM, 30 (April 1983): 820-833.

29. Kurki-Suonio "Notes on Top Down Languages," BIT 9 (1969): 225-238.

30. Leinius, R.P. "Error Detection and Recovery for Syntax Directed Compiler Systems." Ph.D. Dissertation, University of Wisconsin, 1970.

31. Lewis, P.M. and Stearns, R.E. "Syntax Directed Transduction," Journal of ACM 15 (March 1968): 464-488.

32. Martin, D.F. "A Boolean Matrix Method for the Computation of Linear Precedence Functions," Journal of ACM 15 (June 1972): 448-454.

33. Martin, D.F. "Boolean Matrix Methods for the Detection of Simple Precedence Grammars," Communications of ACM 11 (October 1968): 685-687.

34. Mckeeman, W.M.; Horning, J.J.; and Wortman, D.B. A Compiler Generator. Englewood Cliffs, N.J.: Prentice-Hall, 1970.

35. Moll, K.R. "Left Context Precedence Grammars," Acta Informatica 14 (April 1980): 317-336.

36. Naur, P., ed. "Report On the Algorithmic Language ALGOL 60," Communications of ACM 6 (Januray 1963): 1-17.

37. Ogden, W.F. "A helpful result for proving inherent ambiguity," Mathematical Systems Theory 2 (1968): 191-194.

38. Park, J.C.H.; Choe, K.M.; and Chang, C.H. "A New Analysis of LALR(1) Formalisms," ACM Transactions on Programming Languages and Systems 7 (Januray 1985): 615-649.

39. Ripley, G.D. "A Simple Recovery Only Procedure For Simple Precedence Parsers," Communications of ACM 21 (November 1978): 928-930.

40. Rosenkrantz, D.J. and Stearns, R.E. "Properties of Deterministic Top Down Grammars," Information and Control 17 (March 1970): 226-256.

41. Shymasundar, R.K. "Precedence-regular Grammars," International Journal of Computer Mathematics 7 (March 1979): 173-186.

42. Williams, M.H. "Conditions for Extended Operator Precedence Parsing," Computer Journal 22 (Feburary 1979): 164-168.

43. Wirth, N. and Weber, H. "EULER - A Generalization of ALGOL and its Formal Definition, Parts 1 and 2," Communications of ACM 9 (Januray 1966): 89-99.

44. Workman, D.A. "SR(s,k) Parsers: A Class of Shift-Reduce Bounded-Context Parsers," Journal of Computer Systems and Sciences 22 (Feburary 1981): 178-197.

45. Workman, D.A. and Milani, M.T. "Epsilon Weak Precedence Grammars and Languages." Department of Computer Science Technical report CS-TR-68, University of Central Florida, Orlando, Florida: November 1984.

46. Yehudai, A. "A New Definition for Simple Precedence Grammars," BIT 19 (Feburary 1979): 282-284.