INFORMATIQUE THÉORIQUE ET APPLICATIONS

MASOUD T. MILANI
DAVID A. WORKMAN

Epsilon weak precedence grammars and languages

Informatique théorique et applications, tome 24, n° 3 (1990), p. 241-266

<http://www.numdam.org/item?id=ITA_1990__24_3_241_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 24, n° 3, 1990, p. 241 à 266)

EPSILON WEAK PRECEDENCE GRAMMARS AND LANGUAGES (*)

by Masoud T. MILANI (1) and David A. WORKMAN (2)

Communicated by V. GALLIER

Abstract. – *A new class of grammars called Epsilon Weak Precedence (EWP) grammars is obtained by generalizing weak precedence grammars to include ε-rules. We show that the EWP grammars define exactly the deterministic context-free languages and for all $k \geq 0$, the class of EWP grammars with at most k ε-rules define a class of languages that is properly included in the class of languages defined by EWP grammars having no more than $k+1$ ε-rules.*

Résumé. – *Nous définissons une nouvelle classe de grammaires qui sont appellées grammaires de précédence faible avec epsilon (EWP) en généralisant les grammaires de précédence faible dans lesquelles sont permises des ε-règles. Nous démontrons que les grammaires EWP définissent exactement les langages algébriques déterministes. Pour tous $k \geq 0$, la classe de grammaire EWP avec au plus k ε-règles définit une classe de langages qui est proprement inclue dans la classe des langages définie par les grammaires EWP qui ont au plus $k+1$ ε-règles.*

1. INTRODUCTION

Hierarchies of subclasses of context-free languages have been studied in the literature. Kurki-Suonio [12] showed that for all $k \geq 0$, the class of LL(k) grammars defines a class of languages that is properly included in the class of languages defined by LL($k+1$) grammars. Harrison and Havel [7] established a hierarchy of strict deterministic languages that is characterized by the number of states of DPDAs accepting them. Finally, Harrison and Yehudai [8] proved an infinite hierarchy of deterministic context-free languages. They showed that for all $k > 0$, the class of all languages accepted by

(*) Received May 1988, revised in January 1989.
(1) School of Computer Science, Florida International University, University Park, Miami, Florida 33199, U.S.A.
(2) Department of Computer Science, University of Central Florida, Orlando, Florida 32816, U.S.A.

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/90/03 241 26/$4.60/

a DPDA with k accepting configurations is properly included in the class of languages accepted by a DPDA having $k+1$ accepting configurations.

In this paper, we establish a new infinite hierarchy of deterministic context-free languages. We generalize the well-known class of weak precedence grammars by permitting ε-rules. This new class of grammars, that we call the class of Epsilon Weak Precedence (EWP) grammars, is shown to describe exactly the deterministic context-free languages. Our hierarchy of deterministic context-free languages is characterized by the minimum number of ε-rules that must appear in EWP grammars defining them. We show that the class of EWP grammars with at most k ε-rules define a class of languages which is properly included in that defined by EWP grammars having $k+1$ ε-rules.

The remainder of this paper is organized as follows. Section 2 contains our notational conventions, basic definitions and results assumed throughout the paper. Readers familiar with the definitions and notational conventions found in Aho and Ullman [2] may skip this section. The notion of Epsilon Weak Precedence grammars is developed in section 3. In this section, we show that EWP grammars are unambiguous and present the EWP parsing algorithm. Section 4 establishes the equivalence of the EWP and the deterministic context-free languages along with the hierarchy of EWP languages. The paper is completed by our concluding remarks in section 5.

2. PRELIMINARIES

Definitions, examples, lemmas and theorems are numbered sequentially in the order they occur in each section. The number designation has the form $s.k$ where "s" denotes the section number and "k" the occurrence index.

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 $\alpha^n = \alpha\alpha^{n-1}$, $n \geqq 1$. The length of a string, α, is denoted $|\alpha|$. For all $k \geqq 0$, unary operators, $PREF_k$ and $SUFF_k$, are defined on V^* as follows:

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

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

A context-free grammar (abbreviated "CFG") is a 4-tuple, $G=(N, \Sigma, P, S)$, 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. The augmented form of a grammar, $G=(N, \Sigma, P, S)$, is defined as $G'=(N \cup \{ S' \}, \Sigma, P \cup \{ S' \to S \}, S')$, where $S' \notin V_G$. If α and β are strings in V_G^*, we write "$\alpha \Rightarrow_G^{\pi} \beta$" to denote a derivation $\pi \in P^*$ of β from α in G. The notations, "$\Rightarrow^{\pi}$", "$\Rightarrow^*$", and "$\Rightarrow^+$" denote a derivation π, a derivation having zero or more steps and a derivation having one or more steps, respectively. Subscripts *lm* or *rm* are used to denote leftmost or rightmost derivations. If $\rho : A \to \varepsilon$ is a production in P, then A is said to be *erasable*, and ρ is called an ε-rule. The set of erasable nonterminals is denoted N_ε. A nonterminal, Z, is said to be *nullable*, if $Z \Rightarrow^+ \varepsilon$. The set of nullable nonterminals is denoted *null* (G). A string $\alpha\beta\gamma$ is a *sentential form* if $S \Rightarrow^* \alpha\beta\gamma$. 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 $S \Rightarrow^* \alpha A \gamma \Rightarrow \alpha\beta\gamma$. A string $\gamma \in V_G^*$ is called a *viable prefix* of G if $S \Rightarrow_{rm}^* \alpha A \omega \Rightarrow \alpha\beta\omega$ and $\gamma = PREF_k(\alpha\beta)$, for some $k \geqq 0$. For each $X \in V_G$, the set $FIRST_k(X)$, $k \geqq 0$ is defined as

$$FIRST_k(X) = \{ PREF_k(\omega) \mid X \Rightarrow_{lm}^* \omega, \omega \in \Sigma^* \}.$$

A grammar is said to be *uniquely invertible* (UI), if for all A, $B \in N$, $A \to \beta \in P$ and $B \to \beta \in P$ imply $A = B$. A grammar has a *cycle* if there exists $A \in N$ such that $A \Rightarrow^+ A$. A grammar is *reduced* if for each production, $A \to \alpha$, there exist strings $x, y, z \in \Sigma^*$ such that $S \Rightarrow^* xAz \Rightarrow x\alpha z \Rightarrow^* xyz$.

We now review the definitions and the concepts assumed in the paper. The first concept we review is that of simple precedence grammars [15]. We begin with the definition of the precedence relations $\lessdot$, $\doteq$ and $\gtrdot$ defined on the vocabulary of a grammar, G.

DEFINITION 2.1: Let $G=(N, \Sigma, P, S)$ be a CFG without ε-rules. For each $X \in N$ define sets LEFT (X), RIGHT (X) as follows:

$$\text{LEFT}(X) = \{ Z \in V_G \mid X \Rightarrow_G^+ Z\theta, \theta \in V_G^* \},$$
$$\text{RIGHT}(X) = \{ Z \in V_G \mid X \Rightarrow_G^+ \theta Z, \theta \in V_G^* \}.$$

Let $X, Y \in V_G$ and $t \in \Sigma$. The *precedence relations* $\lessdot$, $\doteq$ and $\gtrdot$ are defined on V_G as follows:

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

(2) $X \lessdot Y$, iff there is a $Z \in N$ such that $X \doteq Z$ and $Y \in \text{LEFT}(Z)$;

(3) $X \gtrdot t$, iff there is a $Z \in N$ such that $Z \doteq t$ and $X \in \text{RIGHT}(Z)$ or, there are $Z_1, Z_2 \in N$ where $Z_1 \doteq Z_2$ with $X \in \text{RIGHT}(Z_1)$ and $t \in \text{LEFT}(Z_2)$.

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

DEFINITION 2.2: Let $G=(N, \Sigma, P, S)$ be a CFG. G is said to be *simple precedence* if the following conditions hold.

(1) G has no ε-rules;

(2) G has no cycles;

(3) G is Uniquely Invertible;

(4) The precedence relations, $\lessdot$, $\doteq$ and $\gtrdot$ are pairwise disjoint.

In parsing algorithms based on simple precedence grammars the precedence relations are extended to include a special symbol, *$*, used to denote the end of the input string to be parsed as well as the bottom of the parse stack. The algorithm functions by shifting input symbols onto the stack as long as the relations $\lessdot$ or $\doteq$ hold between the stack top and the next input symbol. A stack reduction is initiated when the relation $\gtrdot$ holds. An error is reported if no relation holds. When stack reductions are made, the relation $\lessdot$ is used to locate the left end of the handle in the stack; the unique invertibility property is then applied to identify the production used to make the stack reduction.

Because a precedence parser always shifts when either $\lessdot$ or $\doteq$ holds between the stack top and the next input, the requirement that $\lessdot$ and $\doteq$ be disjoint can be relaxed. This observation leads to the notion of weak precedence grammar originally due to Ichbiach and Morse [9].

DEFINITION 2.3: A CFG, $G=(N, \Sigma, P, S)$, is said to be *weak precedence* (WP) iff the following conditions hold:

(1) G has no ε-rules;

(2) G is cycle-free;

(3) G is uniquely invertible;

(4) The precedence relations, $\lessdot$, $\doteq$ and $\gtrdot$ satisfy $(\lessdot \cup \doteq) \cap \gtrdot = \varnothing$;

(5) If $X \to \alpha A \beta$ and $Y \to \beta$ belong to P, then $(A, Y) \notin (\lessdot \cup \doteq)$.

In [1] it is established that the weak precedence grammars are equivalent in language power to the class of simple precedence grammars. Fischer [3] has shown the simple precedence languages to be a proper subclass of the deterministic context-free languages by demonstrating that the set $L = \{ a 0^n 1^n \mid n \geqq 0 \} \cup \{ b 0^n 1^{2 \times n} \mid n \geqq 0 \}$ is not a simple precedence language.

Simple precedence grammars can be generalized to (m, n) precedence grammars by increasing the number of symbols used to define the precedence relations [6]. Graham [5] showed that every deterministic context-free language is defined by some (2,1) precedence parsable grammar.

We now review the class of (m, n) Bounded Right-Context grammars [4]. The conditions of (m, n) Bounded Right-Context grammars guarante 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. (1,1) Bounded Right-Context grammars define exactly the class of deterministic context-free languages.

DEFINITION 2.4: A CFG, $G=(N, \Sigma, P, S)$, is said to be (m, n) *Bounded Right-Context* (BRC) for m, $n \geqq 1$, if in the augmented grammar, G', the conditions

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

(2) $\$^m S' \$^n \Rightarrow^*_{rm} \gamma B x \Rightarrow_{rm} \gamma\delta x = \alpha' \beta y$,

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

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

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

We now review the class of LR (1) grammars and parsers due to Knuth [10].

DEFINITION 2.5: A reduced CFG, $G=(N, \Sigma, P, S)$, is said to be LR (k) for $k \geqq 0$, if in the augmented grammar, G', the conditions

(1) $S' \Rightarrow^*_m \alpha A \omega \Rightarrow_{rm} \alpha\beta\omega$,

(2) $S' \Rightarrow^*_{rm} \gamma B x \Rightarrow_{rm} \alpha\beta y$,

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

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

LR grammars define exactly the deterministic context-free languages. We now summarize the behavior of LR parsers. First, the notion of LR (k) items is introduced.

DEFINITION 2.6: Let $G=(N, \Sigma, P, S)$ 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 $S \Rightarrow^*_{rm} \alpha A \omega \Rightarrow_{rm} \alpha\beta_1 \beta_2 \omega$ 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 State Automation (DFSA), known as the GOTO graph. This DFSA recognizes viable prefixes of the underlying grammar. Two functions called ACTION and GOTO 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 behavior 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 $A \rightarrow \alpha$, 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 guarante that every entry of ACTION and GOTO tables is uniquely defined.

3. EPSILON WEAK PRECEDENCE GRAMMARS

Wirth and Weber [15] defined the relations $\lessdot$, $\doteq$ and $\gtrdot$ on the vocabulary of a context-free grammar with no ε-rules by using the notions of LEFT and RIGHT sets. For each nonterminal, X, these sets contain all symbols that could appear, respectively, as the leftmost and rightmost symbols of any sentential form, ω, satisfying: $X \Rightarrow^{+} \omega$. In our first definition we generalize the LEFT and RIGHT sets for grammars containing ε-rules. Specifically, $\mathbf{L}(X)$ corresponds to the LEFT set defined by Wirth and Weber. It is defined in terms of leftmost derivations, $X \Rightarrow_{lm}^{+} \omega$, which do not apply any ε-rule. $\mathbf{L}_\varepsilon(X)$ contains those symbols that can appear leftmost in such derivations that do apply some ε-rule. $\mathbf{R}(\mathbf{X})$ and $\mathbf{R}_\varepsilon(X)$ are the analogs of $\mathbf{L}(X)$ and $\mathbf{L}_\varepsilon(X)$ for right-most derivations.

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

$$\mathbf{L}(X)=\{Z \mid \text{there is a } \pi \in P^{+} \text{ such that } X \Rightarrow_{lm}^{\pi} Z\alpha, Z \in V_G, \alpha \in V_G^{*} \text{ and for all } (\rho: Y \to \varepsilon) \in P,\ \rho \notin \pi\}$$

$$\mathbf{L}_\varepsilon(X)=\{Z \mid \text{there exists } \pi_1 \rho \pi_2 \in P^{+} \text{ such that } (\rho: Y \to \varepsilon) \in P \text{ and } X \Rightarrow_{lm}^{\pi_1} Y\theta \Rightarrow_{lm}^{\rho} \theta \Rightarrow_{lm}^{\pi_2} Z\alpha, Z \in V_G, \theta \in V_G^{+} \text{ and } \alpha \in V_G^{*}\}$$

$$\mathbf{R}(\mathbf{X})=\{Z \mid \text{there is a } \pi \in P^{+} \text{ such that } X \Rightarrow_{rm}^{\pi} \alpha Z, Z \in V_G, \alpha \in V_G^{*} \text{ and for all } (\rho: Y \to \varepsilon) \in P,\ \rho \notin \pi\}$$

$$\mathbf{R}_\varepsilon(\mathbf{X})=\{Z \mid \text{there exists } \pi_1 \rho \pi_2 \in P^{+} \text{ such that } (\rho: Y \to \varepsilon) \in P \text{ and } X \Rightarrow_{rm}^{\pi_1} \theta Y \Rightarrow_{rm}^{\rho} \theta \Rightarrow_{rm}^{\pi_2} \alpha Z, Z \in V_G, \theta \in V_G^{+} \text{ and } V_G^{*}\}$$

The sets $\mathbf{L}$, $\mathbf{L}_\varepsilon$, $\mathbf{R}$ and $\mathbf{R}_\varepsilon$ are used to define the epsilon precedence $<^{+}$, $\stackrel{+}{=}$ and $^{+}>$. The purpose of epsilon precedence relations are similar to that of the precedence relations $\lessdot$, $\doteq$ and $\gtrdot$ (Definition 2.1.) They classify adjacent symbols of viable prefixes of the underlying grammar according to whether they occur at the left end of the handle ($<^{+}$), the right end of the handle ($^{+}>$) or within the handle $\stackrel{+}{=}$. The relations $<^{+}$ and $\stackrel{+}{=}$ are identical to $\lessdot$ and $\doteq$, respectively, if $\mathbf{L}(X)$ is used as the LEFT set of nonterminal X. However, due to the presence of ε-rules, the relation $^{+}>$ is not the same as $\gtrdot$. In addition to pairs of symbols related by $\gtrdot$, the relation $^{+}>$ contains those pairs of symbols, (X, t), that as the result of erasing some part of a

derivation tree constructed entirely of nullable nonterminals, can appear adjacent in a sentential form.

DEFINITION 3.2: The *Epsilon Precedence* (EP) *relations* $<^{+}$, $\stackrel{+}{=}$ and $^{+}>$ for a CFG, G=(N, Σ, P, S), are defined as follows, where X, $Y \in (N \cup \Sigma)$ and $t \in \Sigma$.

(1) $X \stackrel{+}{=} Y$ iff $A \to \alpha XY\beta \in P$.

(2) $X <^{+} Y$ iff there exists $Z \in N$ such that $X \stackrel{+}{=} Z$ and $Y \in \mathbf{L}(Z)$.

(3) Define $X \stackrel{\varepsilon}{=} Y$ if there exist $Z_1 Z_2, \ldots, Z_n$, $n \geqq 1$, in *null*(G) such that $X \stackrel{+}{=} Z_1 \stackrel{+}{=} Z_2 \ldots \stackrel{+}{=} Z_n \stackrel{+}{=} Y$.

Let $\stackrel{*}{=}$ be $\stackrel{\varepsilon}{=} \cup \stackrel{+}{=}$. Then, define $X^{+} > t$ if one of the following conditions holds:

(*a*) $X \stackrel{+}{=} Y$, $Y \in N$, $t \in \mathbf{L}_{\varepsilon}(Y)$;

(*b*) $X \stackrel{\varepsilon}{=} Y$, $t \in \mathbf{L}_{\varepsilon}(Y) \cup \mathbf{L}(Y) \cup \{ Y \}$;

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

(4) Lest \$ be a unique symbol not found in V_G used as the left and right end marker of sentential forms in G. Define:

$\$ <^{+} X$, for all $X \in \mathbf{L}(S)$;

$\$^{+} > t$, for all $t \in \mathbf{L}_{\varepsilon}(S)$;

$X^{+} > \$$, for all $X \in \mathbf{R}_{\varepsilon}(S) \cup \mathbf{R}(S)$;

$\$^{+} > \$$, iff $S \in null(G)$.

Grammars containing more than one ε-rule are not Uniquely Invertible because all the ε-rules have essentially the same empty right-part. In our next definition, the notion of Unique Invertibility is relaxed to apply to grammars having 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 ∈ N, $A \neq B$, $A \to \beta \in P$ and $B \to \beta \in P$ imply $\beta = \varepsilon$.

Our next definition generalizes the class of weak precedence grammars by permitting ε-rules.

DEFINITION 3.4: A CFG, G=(N, Σ, P, S), is said to be an *Epsilon Weak Precedence* (EWP) grammar if:

(1) G is AUI;

(2) G is cycle-free;

(3) $(\stackrel{+}{=} \cup <^{+}) \cap {}^{+}> = \varnothing$;

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

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

$$\rho(Z) = \{ (X,t) \mid (X,Z) \in (\stackrel{\pm}{=} \cup <^+), \\ (X,t) \in {}^+>, (Z,t) \in (\stackrel{\pm}{=} \cup <^+ \cup {}^+>), X \in N \cup \{\$\}, t \in \Sigma \cup \{\$\}\};$$

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

An EWP grammar with k ε-rules is denoted EWP_k and a language **L** is said to be an EWP_k language if and only if it is generated by some EWP_k grammar.

The definition of EWP grammars closely resembles that of WP grammars. Conditions (1)-(4) parallel the definition of WP grammars. Condition (4) also guarantees that a reduction by some ε-rule must apply only when no ethers apply. Condition (5) permits the unique determination of the ε-rule to be used in reducing the stack. This condition is interesting from another view point-it marries the notions of bounded-context and precedence strategies through the vehicle of ε-rules; that is, when "ε" is the handle, the stack top and lookahead become its bounded left and right contexts, respectively. Condition (6) deals with the potential conflicts between stack reduction using ε-rules and parser "acceptance."

The class of languages defined by EWP grammars is larger than that defined by WP grammars. This result is established in our next lemma.

LEMMA 3.5: *The class of WP languages is properly included in the class of EWP languages.*

Proof: Obviously every WP grammar is also an EWP grammar. The EP relations for grammars without ε-rules are identical to Wirth-Weber relations and conditions of Definition 3.4 are satisfied by WP grammars. Hence, the class of WP languages is contained in the class of EWP languages. To show the proper inclusion, consider the language

$$L = \{ a 0^n 1^n \mid n > 0 \} \cup \{ b\, 0^n 1^{2 \times n} \mid n > 0 \}$$

which is known not to be a WP language [3]. Grammar G shown below, is an EWP grammar defining L.

$$\begin{array}{llll} S & \to aX, & S \to bY, \\ X & \to AX1, & Y \to BYC1, \\ X & \to A1, & Y \to BC1, \\ A & \to Z0, & B \to 0, \\ Z & \to \varepsilon, & C \to 1. \end{array}$$

TABLE I

$\mathbf{L}$, $\mathbf{L}_\varepsilon$, $\mathbf{R}$, $\mathbf{R}_\varepsilon$, for G.

	$\mathbf{L}$	$\mathbf{L}_\varepsilon$	$\mathbf{R}$	$\mathbf{R}_\varepsilon$
S	a,b	∅	X,Y,1	∅
X	A,Z	0	1	∅
Y	B,0	∅	1	∅
A	Z	0	0	∅
B	0	∅	0	∅
C	1	∅	1	∅
Z	∅	∅	∅	∅

The sets $\mathbf{L}$, $\mathbf{L}_\varepsilon$, $\mathbf{R}$ and $\mathbf{R}_\varepsilon$ for the nonterminals of G are shown in Table I.

Clearly G is cycle-free and AUI. Moreover, the EP relations for G, shown in Table II, 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 EWP and therefore L is an EWP language.

TABLE II

The EP relations for G.

	S	X	Y	A	B	C	Z	a	b	0	1	$
S												
X											$\doteq$	⋗
Y						$\doteq$					⋖	⋗
A		$\doteq$		⋖			⋖			⋗	$\doteq$	
B			$\doteq$		⋖	$\doteq$				⋖	⋖	
C											$\doteq$	
Z										$\doteq$		
a		$\doteq$		⋖			⋖			⋗		
b			$\doteq$		⋖					⋖		
0										⋗	⋗	
1											⋗	⋗
$								⋖	⋖			

Our next Lemma establishes the fact that if ε is the handle of a right sentential form, αω, then $SUFF_1(\alpha)^+ > PREF_1(\omega)$. This result is latter used to relate the EP relations to derivations in a CFG.

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

$$\$S\$ \Rightarrow^*_{rm} \alpha X \delta a \omega \Rightarrow^+_{rm} \alpha X a \omega, \qquad \text{then} \quad X^+ > a.$$

Proof: Obviously, if either $X=\$$ or $a=\$$, then from Definition 3.2 we have that $X^+ > a$. We therefore assume that $X \neq \$$ and $a \neq \$$. The derivation $\$S\$ \Rightarrow^*_{rm} \alpha X \delta a \omega \Rightarrow^+_{rm} \alpha X a \omega$ may be written as:

$$\$S\$ \Rightarrow^*_{rm} \alpha' A z \Rightarrow_{rm} \alpha' \alpha'' B \delta' Y \beta z \Rightarrow^*_{rm} \alpha' \alpha'' B \delta' Y y z \Rightarrow^*_{rm}$$
$$\alpha' \alpha'' B \delta' \delta'' a x y z = \alpha' \alpha'' B \delta' \delta'' a \omega \Rightarrow^*_{rm} \alpha' \alpha'' B a \omega \Rightarrow^*_{rm}$$
$$\alpha' \alpha'' \theta X \lambda a \omega \Rightarrow^*_{rm} \alpha' \alpha'' \theta X a \omega = \alpha X a \omega,$$

where either $\delta' \delta'' = \delta$ ($X=B$, $\lambda=\varepsilon$, $\theta=\varepsilon$ and $\alpha=\alpha'\alpha''$) or $\lambda=\delta$.

If $\delta=\delta'\delta''$, then $A \to \alpha'' X \delta' Y \beta \in P$ and either $X \stackrel{+}{=} Y$ ($\delta'=\varepsilon$) and $a \in \mathbf{L}_\varepsilon(Y)$, or $X \stackrel{\varepsilon}{=} Y$ ($\delta' \neq \varepsilon$) and $a \in \mathbf{L}_\varepsilon(Y) \cup \mathbf{L}(Y) \cup \{ Y \}$. Hence $X^+ > a$.

If $\delta=\lambda$, then $B \Rightarrow^*_{rm} \theta X \lambda \Rightarrow^+_{rm} \theta X$ and $X \in \mathbf{R}_\varepsilon(B)$. Also $Y \Rightarrow^*_{rm} \delta'' a x \Rightarrow^*_{rm} a x$ and we have that $a \in \mathbf{L}_\varepsilon(Y) \cup \mathbf{L}(Y) \cup \{ Y \}$. Additionally, $A \to \alpha'' B \delta' Y \beta \in P$, $\delta' \Rightarrow^*_{rm} \varepsilon$ and therefore, $B \stackrel{*}{=} Y$. Thus $X^+ > a$.

We now establish that in a right sentential form αβω, with the handle of β, either $\stackrel{+}{=}$ or $<^+$ holds between adjacent symbols of αβ and $SUFF_1(\alpha)^+ > PREF_1(\omega)$. Moreover, we show that if the handle is ε (β=ε) and αω is derived from αZω, then $(SUFF_1(\alpha), PREF_1(\omega)) \in \rho(Z)$.

LEMMA 3.7: Let G=(N, Σ, P, S) be a CFG. If

$$\$S\$ \Rightarrow^n_{rm} X_p X_{p-1} \ldots X_{k+1} A a_1 \ldots a_q \Rightarrow X_p X_{p-1} \ldots X_{k+1} X_k \ldots X_1 a_1 \ldots a_q;$$

then

(1) $(X_{i+1}, X_i) \in (\stackrel{+}{=} \cup <^+)$; $k < i < p$;

(2) $X_1{}^+ > a_1$;

(3) Either $k>0$, and

(3.1) $X_{k+1} <^+ X_k$;

(3.2) $X_{i+1} \stackrel{+}{=} X_i$; $1 \leqq i \leqq k$;

or $k=0$, and

(3.3) $(X_1, a_1) \in \rho(A)$.

Proof: The proof is essentially the same as the proof given in [2] for grammars without ε-rules (An induction on n, the length of the derivation.)

We only observe that if $k=0$ (the handle is ε), conclusion (2) follows from Lemma 3.6 and conclusion (3.3) follows directly from conclusions (1) and (2) and Definition 3.4.

We now show that every EWP grammar is unambiguous. This is achieved by showing that the class of (1,1) Bounded Right-Context grammars properly contains the class of EWP grammars.

LEMMA 3.8: *The class of* (1,1) *BRC grammars properly contains the class of EWP grammars.*

Proof: (1,1) BRC grammars are not necessarily AUI. The class of (1,1) BRC grammars, therefore, do not coincide with the class of EWP grammars. We show that every EWP grammar is also a (1,1) BRC grammar. Assume for the sake of contradiction that $G=(N, \Sigma, P, S)$ is an EWP grammar which is not (1,1) BRC. Then, there must exist derivations

$$S \Rightarrow^*_{rm} \alpha A \omega \Rightarrow_{rm} \alpha\beta\omega \tag{1}$$

$$S \Rightarrow^*_{rm} \gamma B x \Rightarrow_{rm} \gamma\delta x = \alpha' \beta y \tag{2}$$

such that $\omega, x, y \in \Sigma^*$, $|y| \geqq |x|$,

$$SUFF_1(\alpha') = SUFF_1(\alpha), \qquad PREF_1(\omega) = PREF_1(y)$$

but $B \to \delta \neq A \to \beta$. We consider three distinct cases and in each case derive a contradiction for G being EWP.

Case 1. $|x| = |y|$: For this case we must have that either $\delta = \theta\beta$ or $\beta = \theta\delta$, for some $\theta \in V_G^*$. Assume without loss of generality that $\delta = \theta\beta$. Two subcases are considered.

a. $\theta \neq \varepsilon$.

Let $\theta = \theta' X$, $\theta' \in V_G^*$, $X \in V_G$. Then

$$\delta = \theta' X \beta, \qquad \alpha' = \gamma\theta' X \qquad \text{and} \qquad SUFF_1(\alpha) = SUFF_1(\alpha') = X.$$

Moreover, applying Lemma 3.7 to the first derivation, we have that $(X, A) \in (\stackrel{+}{=} \cup <^+)$. But $A \to \beta$ and $B \to \theta' X \beta = \delta$ are productions in P and we have a contradiction for G being EWP.

b. $\theta = \varepsilon$.

If $\theta = \varepsilon$ then $\alpha' = \gamma$ and $\beta = \delta = \varepsilon$ (G is AUI). Applying Lemma 3.7 to derivations (1) and (2), we conclude that

$$(SUFF_1(\alpha), PREF_1(\omega)) \in \rho(A)$$

and

$$(SUFF_1(\gamma), PREF_1(y)) \in \rho(B).$$

But

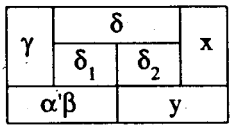
$$SUFF_1(\alpha) = SUFF_1(\alpha') = SUFF_1(\gamma)$$

and

$$PREF_1(\omega) = PREF_1(y).$$

Hence, $\rho(A) \cap \rho(B) \neq \varnothing$ and G is not EWP.

Case 2. $|x| < |y| \leqq |\delta x|$: Assume $\delta = \delta_1 \delta_2$, where $\delta_2 \neq \varepsilon$, $\gamma\delta_1 = \alpha'\beta$ and $\delta_2 x = y$.



Applying Lemma 3.7 to derivations (1) and (2), we have that

$$(SUFF_1(\alpha\beta), PREF_1(\omega)) \in {}^{+}{>}$$

and

$$(SUFF_1(\gamma\delta_1), PREF_1(y)) \in (\stackrel{+}{=} \cup {<}^{+}).$$

But

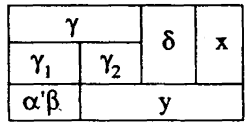
$$SUFF_1(\alpha\beta) = SUFF_1(\alpha'\beta) = SUFF_1(\gamma\delta_1)$$

and

$$PREF_1(\omega) = PREF_1(y).$$

Therefore, $(\stackrel{+}{=} \cup {<}^{+} \cap {}^{+}{>} \neq \varnothing$ and G is not EWP.

Case 3. $|x|<|y|>|\delta x|$: Assume that $y=\gamma_2 \delta x$, $\gamma_2 \neq \varepsilon$. That is, $\gamma=\gamma_1 \gamma_2$, where $\gamma_1=\alpha' \beta$.



Applying Lemma 3.7 to derivations (1) and (2), we have that

$$(SUFF_1(\alpha\beta), PREF_1(\omega)) \in {+>}$$

and

$$(SUFF_1(\gamma_1), PREF_1(\gamma_2)) \in (\stackrel{+}{=} \cup <^+).$$

But

$$SUFF_1(\alpha\beta) = SUFF_1(\alpha' \beta) = SUFF_1(\gamma_1)$$

and

$$PREF_1(\omega) = PREF_1(y) = PREF_1(\gamma_2).$$

Again $(\stackrel{+}{=} \cup <^+) \cap {+>} \neq \varnothing$ and G can not be EWP.

In our next theorem we formally establish that the class of EWP grammars is unambiguous. This result follows directly from Lemma 3.8 and the fact that every (1,1) BRC grammar is unambiguous.

THEOREM 3.9: *Every Epsilon Weak Precedence grammar is unambiguous.*

We conclude this section by presenting our EWP parsing algorithm. The EWP parsing algorithm is essentially the weak precedence parsing algorithm except that here the stack may be reduced by some ε-rule if no other reduction applies. The correctness of the EWP parsing algorithm follows directly from Lemma 3.7 and Definition 3.4.

ALGORITHM 3.10: EWP parsing algorithm.

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

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

METHOD:

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

(*a*) $f(\$\ S,\ \$) = \text{accept}$;

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

$$(X, t) \in (< \cup \stackrel{+}{=}).$$

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

$$(X, t) \in +>.$$

(*d*) $f(X, t) = \text{error}$; otherwise.

(2) For all $X \in V_G \cup \{\$\}$, $t \in \Sigma \cup \{\$\}$ 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 $\alpha = \alpha' \beta$, $\beta \neq \varepsilon$, $i: B \to \beta$ and for all $A \to \delta\beta \in P$, $\delta\beta$ is not a suffix of α;

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

(*c*) $g(\alpha, t) = \text{error}$, otherwise.

4. PROPERTIES OF EWP LANGUAGES

In the previous section we introduced the notion of EWP grammars and showed that the EWP grammars are unambiguous. The purpose of this section is to establish two major properties of EWP languages, namely, the equivalence of EWP and the deterministic context-free languages and the hierarchy of EWP languages. To prove the equivalence of the deterministic context-free and EWP languages, we present an algorithm to automatically transform LR (1) grammars to equivalent EWP grammars.

The algorithm accepts an LR (1) grammar, G, as input and produces an equivalent EWP grammar, G', by encoding the entire GOTO graph of G's LR (1) parser into grammar symbols. This way, a grammar symbol that appears on top of the parse stack will summarize the information that is contained in the entire stack.

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

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

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

METHOD: Let **C** be the canonical collection of LR(1) states for G, such that the states in **C** are labeled $J_0, J_1, \ldots, J_m$, $m > 0$, where J_0 is the initial state in **C**, and for all i, $0 \leqq i \leqq m$, $J_i \notin V_G$.

(1) $\quad S' = [J_0, S], \quad P' = \varnothing, \quad$ and $\quad N' = \{[J_0, S]\}$.

(2) For each state $I \in \mathbf{C}$ and each item $[A \to .A_1 A_2 \ldots A_n \,|\, u] \in I$, $n \geqq 0$, do

Define:

$$I_i = GOTO(I, A_1 A_2 \ldots A_{i-1}) \ (^1), \qquad 1 \leqq i \leqq n+1$$

$$U = \{u \,|\, \textit{there exists an item } [A \to A_1 A_2 \ldots A_n . \,|\, u] \textit{ in } I_{n+1}\}$$

(*a*)

$$N' = N' \cup \{[I_i, A_i] \,|\, 1 \leqq i \leqq n\} \cup \{[I_{n+1}, A \to A_1 A_2 \ldots A_n, U]\} \cup \{I_i^{A_i} \,|\, A_i \in \Sigma, 1 \leqq i \leqq n\}$$

(*b*)

$$P' = P' \cup \{[I, A] \to [I_1, A_1][I_2, A_2] \ldots [I_n, A_n][I_{n+1}, A \to A_1 A_2 \ldots A_n, U]\} \cup \{[I_i, A_i] \to I_i^{A_i} A_i, I_i^{A_i} \to \varepsilon \,|\, A_i \in \Sigma, 1 \leqq i \leqq n\} \cup \{[I_{n+1}, A \to A_1 A_2 \ldots A_n, U] \to \varepsilon\}.$$

Algorithm 4.1 encodes the state information used by the LR(1) perser for G into the symbols of G'. We observe that N' can be written as the union of disjoint sets N'', N_Σ, N_l and N_r defined below:

$$N'' = \{[I, A] \,|\, I \in \mathbf{C}, A \in N \text{ and there exists an item } [A \to .\alpha \,|\, u] \text{ in } I\}$$

$$N_\Sigma = \{[I, a] \,|\, I \in \mathbf{C}, a \in \Sigma \text{ and there exists an item } [A \to \alpha . a \beta \,|\, u] \text{ in } I\}$$

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

$$N_r = \{[I, A \to \alpha, U] \,|\, I \in \mathbf{C}, A \to \alpha \in P, U \subseteq \Sigma \text{ and for all } u \in U \text{ there exists an item } [A \to \alpha . \,|\, u] \text{ in } I\}$$

---

(1) Note that $I_1 = I$.

Moreover, productions in P' have one of the following forms:

- $A \to \varepsilon$, $A \in N_l \cup N_r$
- $A \to Z a$, $A \in N_\Sigma$, $Z \in N_l$ and $a \in \Sigma$
- $[I, A] \to [I_1, A_1][I_2, A_2] \ldots [I_n, A_n][I_{n+1}, A \to A_1 A_2 \ldots A_n, U]$, $n \geqq 0$
  $[I, A] \in N''$, $[I_i, A_i] \in N'' \cup N_\Sigma$, $1 \leqq i \leqq n$, $[I_{n+1}, A \to A_1 A_2 \ldots A_n, U] \in N_r$

and for all i, $1 \leqq i \leqq n+1$ and each $u \in U$, there exists an item $[A \to A_1 A_2 \ldots A_{i-1} . A_i \ldots A_n \mid u]$ in I_i.

We now show that G' in algorithm 4.1 is indeed EWP and $L(G') = L(G)$. First, we establis that $L(G') = L(G)$.

LEMMA 4.2: In Algorithm 4.1, $L(G') = L(G)$.

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

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

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

BASIS: $n = 1$. For $n = 1$, $A \to \omega$ is a production in P. Moreover, for each $I \in \mathbf{C}$ such that $[I, A] \in N'$, there must exist a production

$$[I, A] \to [I_1, \omega_1][I_2, \omega_2] \ldots [I_m, \omega_m][I_{m+1}, A \to \omega_1 \omega_2 \ldots \omega_m, U]$$

in P', where $\omega = \omega_1 \omega_2 \ldots \omega_n$ and

$$[I_{m+1}, A \to \omega_1 \omega_2 \ldots \omega_m, U] \to \varepsilon$$
$$[I_i, \omega_i] \to I_i^{\omega_i} \omega_i$$
$$I_i^{\omega_i} \to \varepsilon$$

are productions in P'. Hence,

$$\begin{aligned}
[I, A] &\Rightarrow [I_1, \omega_1][I_2, \omega_2] \ldots [I_m, \omega_m][I_{m+1}, A \to \omega_1 \omega_2 \ldots \omega_m, U] \\
&\Rightarrow [I_1, \omega_1][I_2, \omega_2] \ldots [I_m, \omega_m] \\
&\Rightarrow [I_1, \omega_1][I_2, \omega_2] \ldots [I_{m-1}, \omega_{m-1}] I_m^{\omega_m} \omega_m \\
&\Rightarrow [I_1, \omega_1][I_2, \omega_2] \ldots [I_{m-1}, \omega_{m-1}] \omega_m \\
&\Rightarrow^{*} \omega_1 \omega_2 \ldots \omega_{m-1} \omega_m
\end{aligned}$$

That is, $[I, A] \Rightarrow_G^* \omega$.

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

$$A \Rightarrow_G A_1 A_2 \ldots A_m \Rightarrow_G^{n-1} \omega_1 \omega_2 \ldots \omega_m = \omega.$$

That is, $A \to A_1 A_2 \ldots A_m$, $m>0$, is a production in P, and for all i, $1 \leqq i \leqq m$, $A_i \Rightarrow_G^{n_i} \omega_i$, $n_i<n$, $\omega_i \in \Sigma^*$. Moreover, for all $I \in \mathbf{C}$ such that $[I, A] \in N'$, P' must contain a production

$$[I, A] \to [I_1, A_1][I_2, A_2] \ldots [I_m, A_m][I_{m+1}, A \to A_1 A_2 \ldots A_m, U].$$

Obviously, for each i, $1 \leqq i \leqq m$, such that $A_i \in \Sigma$, we have that $A_i = \omega_i$ and

$$[I_i, A_i] = [I_i, \omega_i] \Rightarrow I_i^{\omega_i} \omega_i \Rightarrow \omega_i = A_i$$

is a derivation in G'. Moreover, for all i, $1 \leqq i \leqq m$, such that $A_i \in N$, we have that $A_i \Rightarrow_G^{n_i} \omega_i$, $n_i<n$ and it follows from the inductive hypothesis that $[I_i, A_i] \Rightarrow_G^* \omega_i$. Hence,

$$\begin{aligned}[I, A] &\Rightarrow [I_1 A_1][I_2, A_2] \ldots [I_m, A_m][I_{m+1}, A \to A_1 A_2 \ldots A_m, U] \\ &\Rightarrow [I_1, A_1][I_2, A_2] \ldots [I_m, A_m] \\ &\Rightarrow^* \omega_1 \omega_2 \ldots \omega_{m-1} \omega_m = \omega\end{aligned}$$

is a derivation in G' and the desired result is established.

Only if: Suppose $[I, A] \Rightarrow_G^* \omega$. Define the homomorphism h as follows:

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

Now consider a derivation

$$[I, A] = \alpha_0 \Rightarrow_{rm} \alpha_1 \Rightarrow_{rm} \alpha_2 \ldots \Rightarrow_{rm} \alpha_n$$

in G'. By an induction on n, we show that $A \Rightarrow h(\alpha_n)$ is a derivation in G.

BASIS: $n=0$. For $n=0$, we have that $\alpha_0 = [I, A]$ and $h(\alpha_0) = A$. Clearly, $A \Rightarrow^* h(\alpha_0) = A$ is a derivation of length zero in G.

INDUCTIVE STEP: Assume that for all k, $0 \leq k < n$, $n > 0$, $A \Rightarrow_{rm}^{*} h(\alpha_k)$ is a derivation in G, and consider the derivation

$$[\mathrm{I}, \mathrm{A}] \Rightarrow_{rm}^{n-1} \alpha_{n-1} = \alpha'_{n-1} X \nu \Rightarrow \alpha'_{n-1} \beta \nu = \alpha_n.$$

That is, $X \to \beta$ is the production used to derive α_n from α_{n-1}. If $\beta = \varepsilon$, then clearly

$$h(\alpha_{n-1}) = h(\alpha'_{n-1} X \nu) = h(\alpha'_{n-1}) h(X) h(\nu) = h(\alpha'_{n-1}) h(\nu) = h(\alpha'_{n-1} \nu) = h(\alpha_n).$$

On the other hand, if $\beta \neq \varepsilon$, then $X = [J, B]$, $J \in \mathbf{C}$ and $B \in N \cup \Sigma$. If $B \in \Sigma$, then we must have that $\beta = J^B B$ and similar to the previous case we have that $h(\alpha_{n-1}) = h(\alpha_n)$. However, if $B \in N$, then

$$\beta = [J_1, B_1][J_2, B_2] \ldots [J_l, B_l][J_{l+1}, B \to B_1 B_2 \ldots B_l, U]$$

and

$$B \to B_1 B_2 \ldots B_l \in P.$$

Hence,

$$h(\alpha_{n-1}) = h(\alpha'_{n-1} [J, B] \nu) = h(\alpha'_{n-1}) h([J, B]) h(\nu) = h(\alpha'_{n-1}) B \nu$$

and

$$\begin{aligned} h(\alpha_n) &= h(\alpha'_{n-1} \beta \nu) = h(\alpha'_{n-1}) h(\beta) h(\nu) \\ &= h(\alpha'_{n-1}) h([J_1, B_1][J_2, B_2] \ldots [J_l, B_l][J_{l+1}, B \to B_1 B_2 \ldots B_l, U]) h(\nu) \\ &= h(\alpha'_{n-1}) B_1 B_2 \ldots B_l \nu. \end{aligned}$$

Thus, $h(\alpha_{n-1}) \Rightarrow^{*} h(\alpha_n)$ in G, and it follows from the inductive hypothesis that $A \Rightarrow^{*} h(\alpha_n)$ is a derivation in G.

The special case of derivation (1) where I is the label of the initial state in $\mathbf{C}$ and $A = S$ results in the statement

$$S' \Rightarrow_{G}^{*} \omega \quad \text{if and only if} \quad \mathrm{S} \Rightarrow_{G}^{*} \omega, \ \omega \in \Sigma^{*}$$

proving $L(G) = L(G')$.

We now prove that G' of algorithm 4.1 is EWP.

LEMMA 4.3: In Algorithm 4.1, G' is an EWP grammar.

Proof: We prove that G' is an EWP grammar by showing that G' satisfies the conditions of Definition 3.4.

1. G' is Almost Uniquely Invertible.

Let $A \to \alpha$ and $B \to \alpha$, $\alpha \neq \varepsilon$ be productions in P'. We show that $A = B$. Since $\alpha \neq \varepsilon$, then exactly one of the following must hold:

a. $\alpha = I^a\, a$, $I \in \mathbf{C}$ and $a \in \Sigma$.

b. $\alpha = [I_1, X_1][I_2, X_2] \ldots [I_m, X_m][I_{m+1}, X \to X_1 X_2 \ldots X_m, U]$, $m \geqq 0$.

Clearly, if (*a*) above holds, then $A = [I, a] = B$. If the condition (*b*) holds, howerver, we must have that $A = [I_1, X]$ and $B = [I_1, X']$, $X, X' \in N$ and for all $u \in \mathrm{U}$ the items

$$[X \to X_1 X_2 \ldots X_m . \mid u]$$
$$[X' \to X_1 X_2 \ldots X_m . \mid u]$$

belong to state I in **C**. But G is LR(1). Therefore, these items are not distinct and $X = X'$. Hence G' is AUI.

2. G' is cycle-free.

Assume the contrary and let $[I, A] \Rightarrow^+ [I, A]$ be a derivation in G'. Obviously, $A \notin \Sigma$. Applying an argument similar to the one provided for Lemma 4.2, we conclude that $A \Rightarrow^+ A$ is a derivation in G. But G is assumed to be LR(1) and therefore, cycle-free. Hence, G' is cycle-free.

3. $(\overset{+}{=} \cup <^+) \cap {}^+> = \varnothing$ for G'.

If $(\overset{+}{=} \cup <^+) \cap {}^+> \neq \varnothing$ for G', then there must exist $A \in V_{G'}$ and $b \in \Sigma$ (2) such that either $(A, b) \in (\overset{+}{=} \cap {}^+>)$ *or* $(A, b) \in (<^+ \cap {}^+>)$. 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. *$A <^+ b$ and $A^+ > b$*: If $A <^+ b$, then there must exist a production $X \to \alpha A B \beta$ in P', where $b \in \mathbf{L}(B)$. But each terminal symbol, b, appears in productions of the form $[I, b] \to I^b\, b$, where $I^b \to \varepsilon$ is the only production with I^b in its left-part. Thus, for all $Y \in N'$, $\mathbf{L}(Y) \cap \Sigma = \varnothing$ and therefore, for no b in Σ, the relation $A <^+ b$ may hold. Hence, $(<^+ \cap {}^+>) = \varnothing$ for G'.

CASE 2. *$A \overset{+}{=} b$ and $A^+ > b$:* If $A \overset{+}{=} b$, then clearly A is a symbol of the form I^b, $I \in \mathbf{C}$. The symbol I^b appears only in the production $[I, b] \to I^b\, b$ and therefore is related to b only by $\overset{+}{=}$. Thus, $(\overset{+}{=} \cap {}^+>) = \varnothing$ for G'.

4. If $A \to \alpha X \beta$, and $\mathrm{B} \to \beta$ are productions in P', then $(X, B) \notin (\overset{+}{=} \cup <^+)$.

Assume the contrary and let $A \to \alpha X \beta$ and $B \to \beta$ be distinct productions in P' where $(X, B) \in (\overset{+}{=} \cup <^+)$. Clearly, $\beta \neq \varepsilon$. Otherwise, we must have that

(2) Observe that if $(X, Y) \in {}^+>$, then $Y \in \Sigma$.

$X \in \Sigma \cup N_r$ and since the symbols in $\Sigma \cup N_r$ appear only in the extreme right of any right-part in G', $(X, B) \notin (\stackrel{\pm}{=} \cup <^+)$. Let $\beta = \beta' Y$. It then follows from the forms of the productions in P' that either $Y \in \Sigma$ or $Y = [I, C \to \delta, U] \in N_r$. If $Y \in \Sigma$, then we must have that $\beta' = I^Y \in N_l$ for some $I \in \mathbf{C}$ and therefore $\alpha X = \varepsilon$. This must hold since the symbolds of N_l appear only in the extreme left of any right-part in G'. On the other hand if $Y = [I, C \to \delta, U] \in N_r$, then it follows from the construction that $|\alpha X \beta'| = |\beta'| = |\delta|$. Again αX must be the empty string. Considering the fact that G' is AUI, we conclude that $A \to \alpha X \beta$ and $B \to \beta$ are not distinct and the proof is complete.

5. For all $Z_1, Z_2 \in N_\varepsilon(G')$, $Z_1 \neq Z_2$, $\rho(Z_1) \cap \rho(Z_2) = \varnothing$.

Let $(X, t) \in \rho(Z_1) \cap \rho(Z_2)$. We show that $Z_1 = Z_2$. Clearly $X \in N'' \cup \{\$\}$ and Z_1 and Z_2 belong to $N_r \cup N_l$. Three cases are considered:

CASE 1. $Z_1 \in N_l$ and $Z_2 \in N_l$: Obviously, each symbol of the form I^a in N_l appears only in a unique production, $[I, a] \to I^a a$ and for each $(Y, b) \in \rho(I^a)$, we must that $b = a$ and $Y <^+ I^a$. Thus, $Z_1 = J_1^t$, $Z_2 = J_2^t$, $X <^+ J_1^t$ and $X <^+ J_2^t$, J_2, $J_2 \in \mathbf{C}$. If $X = \$$, then $Z_1 = Z_2 = J_0^t$ where J_0 is the label of the initial state in $\mathbf{C}$. If $X \in N''$, then there must exist productions

$$[K, A] \to [K_1, A_1][K_2, A_2] \ldots [K_n, A_n][K_{n+1}, A \to A_1 A_2 \ldots A_n, U]$$

and

$$[K', A'] \to [K'_1, A'_1][K'_2, A'_2] \ldots [K'_m, A'_m][K'_{m+1}, A' \to A'_1 A'_2 \ldots A'_m, U']$$

such that for some i and j, $1 \leqq i < n$, $1 \leqq j < m$,

$$[K_i, A_i] = [K'_j, A'_j] = X,$$
$$J_1^t \in \mathbf{L}([K_{i+1}, A_{i+1}])$$

and

$$J_2^t \in \mathbf{L}([K'_{j+1}, A'_{j+1}]).$$

That is,

$$K_i = K'_j,$$
$$A_i = A'_j,$$
$$K_{i+1} = J_1$$

and

$$K'_{j+1} = J_2.$$

But, $K_{i+1} = \text{GOTO}(K_i, A_i) = \text{GOTO}(K'_j, A'_j) = K'_{j+1}$, Thus, $J_1 = J_2$ and therefore, $J'_1 = J'_2$. Hence, $Z_1 = Z_2$.

CASE 2. $Z_1 \in N_r$ and $Z_2 \in N_r$: Let $Z = [I_{n+1}, A \to A_1 A_2 \ldots A_n, U] \in N_r$. Then there exists a production

$$[I, A] \to [I_1, A_1][I_2, A_2] \ldots [I_n, A_n][I_{n+1}, A \to A_1 A_2 \ldots A_n, U]$$

in P'. Assume that $(Y, b) \in \rho(Z)$. Obviously, $Y \in N'' \cup \{\$\}$ and $b \in \Sigma \cup \{\$\}$. If $Y = \$$, then we must have that $n = 0$ and $[I, A] \in \mathbf{L}([J_0, S]) \cup \{[J_0 \cdot S]\}$ where $[J_0, S] = S'$, and therefore $I_{n+1} = J_0$. On the other hand if $Y \neq \$$, then either

- $n > 0$ and $Y = [I_n, A_n]$; or
- $n = 0$ and there exists a production

$$[K, B] \to [K_1, B_1][K_2, B_2] \ldots [K_m, B_m][K_{m+1}, B \to B_1 B_2 \ldots B_m, V]$$

such that $m > 1$, $[I, A] \in \mathbf{L}([K_{i+1}, B_{i+1}])$ and $Y = [K_i, B_i]$, $0 < i < m$. Hence,

$$I_{n+1} = \begin{cases} J_0 & Y = \$; \\ GOTO\ (K', Y') & Y = [K', Y'] \end{cases}$$

Moreover, $Z^+ > b$ and we must have that either $[I, A] = [J_0, S]$ or there is a production

$$[K', B'] \to [K'_1, B'_1][K'_2, B'_2] \ldots [K'_l, B'_l][K'_{l+1}, B' \to B'_1 B'_2 \ldots B'_l, V],$$

such that $l > 1$ and for some i, $1 \leqq i < l$, we have that

$$[\mathbf{I}, \mathbf{A}] \in \mathbf{R}_\varepsilon([K'_i, B'_i]) \cup \{[K'_i, B'_i]\}$$
$$b \in FIRST_1([K'_{i+1}, B'_{i+1}][K'_{i+2}, B'_{i+2}] \ldots [K'_l, B'_l]) \cup \{\$\}.$$

Therefore, there exists an item $[A \to A_1 A_2 \ldots A_n . \mid b]$ in state I_{n+1} of $\mathbf{C}$.

Now let $Z_1 = [I_1, A_1 \to \alpha_1, U_1]$ and $Z_2 = [I_2, A_2 \to \alpha_2, U_2]$ belong to N_r and let $(X, b) \in \rho(z_1) \cap \rho(z_2)$. Then $I_1 = I_2$ and we must have that $A_1 \to \alpha_1 = A_2 \to \alpha_2$. Otherwise, $[A_1 \to \alpha_1 . \mid b]$ and $[A_2 \to \alpha_2 . \mid b]$ are two distinct reduce items in state $I_1 = I_2$ in $\mathbf{C}$ and G is not LR(1). Moreover, if follows from the construction that $U_1 = U_2$. Hence, $Z_1 = Z_2$.

CASE 3. $Z_1 \in N_r$ and $Z_2 \in N_l$: Employing arguments similar to those provided for cases 1 and 2 we can show that there must exist two items

$[A \to \alpha . \mid b]$ and $[B \to \beta . b\delta \mid t]$ in the same state in **C**. Again, we have a contradiction to the assumption of G being LR(1).

A direct result of Lemmas 4.2 and 4.3 and Theorem 3.9 is that the class of EWP grammars describes exactly the deterministic context-free grammars.

THEOREM 4.4: *The class of EWP languages is equivalent to the class of deterministic context-free languages.*

Fisher [3] showed the simple precedence languages to be a proper subclass of the deterministic context-free languages by demonstrating that the set $L = \{ a0^n 1^n \mid n \geqq 1 \} \cup \{ b0^n 1^{2 \times n} \mid n \geqq 1 \}$ is not a simple precedence language. As our final theorem we generalize Fischer's result by establishing an infinite hierarchy of EWP languages. This is done by showing that for each $k \geqq 1$, the set $L_k = \bigcup_{i=1}^{k+1} \{ a_i 0^n 1^{i \times n} \mid n \geqq 1 \}$, is an EWP_k language that is not an EWP_{k-1} language.

The intuitive idea behind the proof is the fact that the precedence parsers, in general, do not fully utilize the left-context information that is contained in the parse stack. Any EWP parser acting upon a sentence of L_k must remember the sentence's first symbol as well as the mumber of zeros that is contains. Otherwise, zeros and ones can not be properly matched. Since the stack is the only memory available to the parser, this information must somehow be stored in the stack. The information regarding the first symbol can be propagated to the top of the stack through ε-reductions that for each possible beginning put a different symbol on stack top (*see* the grammar in Lemma 3.5). Therefore, if there are $k+1$ possible beginnings, then there must be at least k different erasable symbols. Otherwise, there would be at least two different sentence beginnings that would propagate to the stack top as if they were the same. Consequently, the parser would not know how to match zeros and ones.

THEOREM 4.5: *For all $k \geqq 0$, the class of EWP_k languages forms a proper subclass of the EWP_{k+1} languages.*

Proof: Clearly, every EWP_k language is also EWP_{k+1}. Let $L_k = \bigcup_{i=1}^{k+1} \{ a_i 0^n 1^{i \times n} \mid n \geqq 1 \}$, for all $k \geqq 1$. For each $k \geqq 1$ a grammar similar that given in Lemma 3.5 will establish tat L_k is an EWP_k language. We will show that any EWP_{k-1} grammar that derives sentences of L_k also derives sentences not in L_k. It is well known [3] that L_1 is not a simple precedence language and thus not EWP_0, therefore we assume that $k > 1$.

Suppose $G=(N, \Sigma, P, S)$ is an EWP_{k-1} grammar such that $L(G)=L_k$. Also assume that an input $a_i 0^n \omega$, $\omega \in 1^*$, $n \geqq 1$, $1 \leqq i \leqq k+1$ is to be parsed by the Algorithm 3.10. Let $\alpha_i(j)$ denote the stack contents after the j^{th} zero of $a_i 0^n \omega$ is shifted. Following an argument similar to that given in [2] showing L_1 is not simple precedence, it can be shown that each zero of $a_i 0^n \omega$ is eventually shifted onto the stack (if not, then the parser can not count zeros.) Furthermore, there exist j_i, l_i, β_i, $0 \leqq j_i \leqq l_i$, $l_i > 0$, $\beta_i \in V_G^+$, satisfying

1. $\alpha_i(j_i + m \times l_i) = \alpha_i(j_i)\, \beta_i^m$, for all $m \geqq 0$ such that $j_i + m \times l_i < n$;
2. β_i is not a suffix of $\alpha_i(j_i)$;
3. β_i is the shortest string in V_G^+ satisfying (1) and (2).

We now show that there exist p and q, $1 \leqq p < q \leqq k+1$ such that for all $r \geqq 0$ if after reading $a_p 0^{j_p + r}$, the stack contains $\alpha_p(j_p)\,\delta$, then after reading $a_q 0^{j_q + r}$ the stack will contain $\alpha_q(j_q)\,\delta$ (*i.e.* $\beta_p = \beta_q$, and $l_p = l_q$).

For all $1 \leqq i \leqq k+1$, with $\alpha_i(j_i)$ on the stack and zero on the input, the parser has exactly k distinct choices. It can either shift the zero or reduce by one of the $k-1$ ε-rules (no other reduction is possible, because no symbol of $\alpha_i(j_i)$ can participate in any reduction). Therefore there must exist p and q, $1 \leqq p < q \leqq k+1$ such that the action performed by the parser with $\alpha_p(j_p)$ on the stack and zero on input, is the same as the action taken with $\alpha_q(j_q)$ on the stack and zero on input. If the action performed is reduce by some ε-rule, then after zero or more additional reductions not involving symbols of $\alpha_p(j_p)$ or $\alpha_q(j_q)$, the input zero must be shifted. Therefore in both cases immediately after shifting the input zero the stack must contain $\alpha_i(j_i)\,\delta 0$, $i \in \{p, q\}$, $\delta \Rightarrow^* \varepsilon$. From this point on as long as zero's appear on input, the action performed by the parser must be the same for p and q. This is true because in both cases the input symbol and the stack symbols that can participate in any reduction are the same. It follows, therefore, that $\beta_p = \beta_q$ and $l_p = l_q$.

Now consider the moves made by the parser acting on the input string $a_p 0^{j_p + m \times l + (l - j_p)}\, 1^{p \times (j_p + m \times l + (l - j_p))}$, where $l = l_p = l_q$. After reading $a_p 0^{j_p + m \times l + (l - j_p)}$, the stack will contain $\alpha_p(j_p)\,\beta^m\,\delta$, $\delta \Rightarrow^* 0^{l - j_p}$. Let s be the largest integer such that after reading $1^{p \times (j_p + m \times l + (l - j_p)) - s}$ the parser will have $\alpha_p(j_p)\,\xi$ on its stack for some $\xi \in V_G^*$ (the nest reduction involves at least one symbol of $\alpha_p(j_p)$). Similarly, consider the input $a_q 0^{j_q + m \times l + (l - j_p)}\, 1^{q \times (j_q + m \times l + (l - j_p))}$. After reading $a_q 0^{j_q + m \times l + (l - j_p)}$, the stack will contain $\alpha_q(j_q)\,\beta^m\,\delta$. Let r be the largest integer such that after reading $1^{q \times (j_q + m \times l + (l - j_p)) - r}$ the parser will have $\alpha_q(j_q)\,\xi$ on the stack. Now consider the string $a_q 0^{j_q + m \times l + (l - j_p)}\, 1^{p \times (j_p + m \times l + (l - j_p)) - s + r}$. $a_q 0^{j_q + m \times l + (l - j_p)}$ causes the stack to become $\alpha_q(j_q)\,\beta^m\,\delta$ and $1^{p \times (j_p + m \times l + (l - j_p)) - s}$ causes the stack to

become $\alpha_q(j_q)\xi$ and the next 1^r causes the acceptance. But for $a_q 0^{j_q+m\times l+(l-j_p)} 1^{p\times(j_p+m\times l+(l-j_p))-s+r}$ to belong to L_k, we must have $q\times(j_q+m\times l+(l-j_p))=p\times(j_p+m\times l+(l-j_p))-s+r$, for all $m\geqq 0$. This equality, however, may hold only when $m=0$. Thus G is not EWP_{k-1}.

5. CONCLUSION

In this paper, we have extended the class of weak precedence grammars by permitting ε-rules. The resulting class of grammars, we have called Epsilon Weak Precedence grammars (EWP), has been shown to possess two important porperties: they define exactly the deterministic context-free languages and for each $k\geqq 0$, the class of languages defined by EWP grammars with at most k ε-rules is properly included in the class of languages defined by EWP grammars with at most $k+1$ ε-rules.

The reason why ε-rules increase the language power of weak precedence grammars is that the erasable symbols can be used to encode the critical context-information (that are similar to those recorded in the states of LR (1) parsers) into the right-parts of the productions. A reduction by some ε-rule in an EWP parser is similar to a state transition in an LR(1) parser. The number of ε-rules that are required to define a languages by an EWP grammar, in this sense, relates to the number of states that any LR type parser for the languages must have.

Finally, we would like to point out that our EWP parsing strategy is similar to the Mixed Strategy Precedence technique [13] in that Reduce/Reduce conflicts are resolved using a bounded context information. Reduce/Reduce conflicts in our scheme, however, may arise only when the handle is the empty string.

REFERENCES

1. A. V. Aho, P. J. Denning and J. D. Ullman, *Weak and Mixed Strategy Parsing*, J. A.C.M., Vol. 19, (2), 1972, pp. 225-243.
2. A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation and Compiling*: Vols. 1 and 2, Prentice-Hall, Englewwod Clifffs, N.J., 1972.
3. M. J. Fischer, *Some Properties of Precedence Languages*, Proc. A.C.M. Symposium on Theory of Computing, 1969, pp. 181-190.
4. R. W. Floyd, *Bounded Context Syntactic Analysis*, Comm. A.C.M., Vol. 7, (2), 1964, pp. 62-67.

5. S. L. GRAHAM, *Extended Precedence Languages, Bounded Right Context Languages and Deterministic Languages*, I.E.E.E. Conference Record of the 11th Annual Symposium on Switching and Automata Theory, 1970, pp. 175-180.
6. J. N. GRAY, *Precedence Parsers for Programming Languages*, Ph. D. Dissertation, Dept. of Computer Science, Univ. of California, Berkeley, 1969.
7. M. A. HARRISON and I. M. HAVEL, *Strict Deterministic Languages*, J. Comput. Syst. Sci., Vol. 7, 1973, pp. 237-277.
8. M. A. HARRISON and A. YEHUDAI, *A Hierarchy of Deterministic Languages*, J. Comput. Syst. Sci., Vol. 19, 1979, pp. 63-78.
9. J. D. ICHBIAH and S. P. MORSE, *A Technique for Generating Almost Optimal Floyd-Evans Productions for Precedence Grammars*, Comm. A.C.M., Vol. 13, (8), 1970, pp. 501-508.
10. D. E. KNUTH, *On the Translation of Languages from Left to Right*, Info. Contr., Vol. 8, (6), pp. 607-639.
11. Y. KREVNER and A. YEHUDAI, *An Iteration Theorem for Simple Precedence Languages*, J.A.C.M., Vol. 30, (4), 1983, pp. 820-833.
12. R. KURKI-SUONIO, *Note on Top Down Languages*, B.I.T., Vol. 9, 1969, pp. 225-238.
13. W. M. MCKEEMAN, J. J. HORNING and D. B. WORTMAN, *A Compiler Generator*, Prentice-Hall, Englewood Cliffs, N.J., 1970.
14. A. NIJHOLT, *Parsing Strategies: A Concise Survey*, Mathematical Foundations of Computer Science, LNCS 118, 1981, pp. 103-120 Springer Berlin.
15. N. WIRTH and H. WEBER, *EULER-A Generalization of ALGOL and its Formal Definition*, Parts 1 and 2, Comm. A.C.M., Vol. 9, (1), 1966, pp. 13-23 and Vol. 9, (2), 1966, pp. 89-99.