SOME PROPERTIES OF PRECEDENCE LANGUAGES

Michael J. Fischer

Carnegie-Mellon University
Pittsburgh, Pennsylvania

Summary

The classes of languages definable by operator precedence grammars[1] and by Wirth-Weber precedence grammars[2] are studied. A grammar is backwards-deterministic[3] if no two productions have the same right part. Operator precedence grammars have no more generative power than backwards deterministic operator precedence grammars, but Wirth-Weber precedence grammars (i.e., grammars having unique Wirth-Weber precedence relations) are more powerful than backwards-deterministic Wirth-Weber precedence grammars; indeed they can generate any context-free language. An algorithm is developed for finding a Wirth-Weber precedence grammar equivalent to a given operator precedence grammar, a result of possible practical significance. The operator precedence languages are shown to be a proper subclass of the backwards-deterministic Wirth-Weber precedence languages which in turn are a proper subclass of the deterministic context-free languages.

1. Introduction

Syntactic analyzers utilizing the notion of "precedence" or "hierarchy" of operators to parse arithmetic expressions in programming languages were in use before much of the present theory of context-free languages had been developed. In 1963, Floyd defined operator precedence grammars as a formalization of the intuitive notion of precedence.[1] In 1966, Wirth and Weber extended the definition of precedence relations to include the non-terminal as well as the terminal symbols of the grammar.[2] However, the details of their definition are somewhat different from Floyd's, and it happens that there are grammars which are precedence by Floyd's definition but not by Wirth and Weber's.

In this paper, we study the classes of languages generated by operator precedence and by Wirth-Weber precedence grammars. The two types of grammars are defined precisely in section 2.

Wirth and Weber present a parsing algorithm for their precedence languages.[2] It is deterministic only if the grammar has unique right parts, that is, no two rules have the same right hand side. Following the definitions of McNaughton,[3] we call such a grammar backwards-deterministic. We show in section 3 that without the requirement that the grammar be backwards-deterministic, we can find a Wirth-Weber precedence grammar for any context-free language, so no deterministic analyzer can work for all Wirth-Weber precedence grammars.

On the other hand, we show that for every operator precedence grammar, we can find an equivalent backwards-deterministic operator precedence grammar. This grammar can be used to drive a deterministic pushdown store recognizer similar to Wirth and Weber's parser, showing that the operator precedence languages are indeed deterministic context-free. We remark that the algorithm which Floyd gives in his paper[1] is a parser for strings in the language but is not a recognizer--it will also parse non-sentences.

Although there are grammars which are operator precedence but not backwards-deterministic Wirth-Weber precedence (hereafter abbreviated BDWWP), we establish in section 4 that Wirth and Weber's definitions are a true generalization of Floyd's in the sense that every operator precedence language is generated by some BDWWP grammar which can be found effectively from the operator precedence grammar. Moreover, there are BDWWP languages which cannot be generated by any operator precedence grammar.

The BDWWP languages do not exhaust the class of deterministic context-free languages, for we exhibit a language which has no BDWWP grammar but is recognizable by a deterministic pushdown store machine with only one state! On the other hand, there are operator precedence languages which require more than one state for recognition by a deterministic pushdown store machine, so we conclude that the one-state deterministic languages are setwise incomparable with both the operator precedence languages and with the BDWWP languages. Hence all three classes are properly contained in the class of deterministic context-free languages.

2. Definitions

A context-free grammar is a quadruple $G = (V, V_T, S, P)$ where V is a finite vocabulary, $V_T \subseteq V$ is the set of terminal symbols, $S \subseteq V\text{-}V_T$ is the start symbol, and $P \subseteq (V\text{-}V_T) \times V^+$ is a set of productions.* We generally write an element $\langle A,\sigma\rangle$ of P as $A \to \sigma$. We call $V_N = V\text{-}V_T$ the set of non-terminal symbols of the grammar. Given $\alpha,\beta \in V^+$, we say that $\alpha \underset{G}{\Rightarrow} \beta$ if there exists $\gamma_1,\gamma_2 \in V^*$ and a production $A \to \sigma$ in P such that $\alpha = \gamma_1 A \gamma_2$ and $\beta = \gamma_1 \sigma \gamma_2$. We say that $\alpha \underset{G}{\overset{n}{\Rightarrow}} \beta$, $n \geq 0$, if there exist $\sigma_0,\ldots,\sigma_n \in V^+$ such that $\alpha = \sigma_0 \underset{G}{\Rightarrow} \sigma_1 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} \sigma_n = \beta$. We say $\alpha \underset{G}{\overset{+}{\Rightarrow}} \beta$ if $\alpha \underset{G}{\overset{n}{\Rightarrow}} \beta$ for some $n \geq 1$, and $\alpha \underset{G}{\overset{*}{\Rightarrow}} \beta$ if

---

* $V^+ = VV^*$.

$\alpha \overset{n}{\underset{G}{\Rightarrow}} \beta$ for some $n \geq 0$. The language generated by G is $L(G) = \{w \in V_T^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$. Two grammars G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$.

A grammar is backwards-deterministic if it has the property that no two productions have the same right part, that is if $A \to \sigma \in P$ and $B \to \sigma \in P$, then $A = B$.

For any context-free grammar $G = (V, V_T, S, P)$, we define the left and right sets of a symbol $A \in V_N = V - V_T$:

$$\mathcal{L}_G(A) = \{B \in V \mid A \overset{+}{\underset{G}{\Rightarrow}} B\tau \text{ for some } \tau \in V^*\};$$

$$\mathcal{R}_G(A) = \{B \in V \mid A \overset{+}{\underset{G}{\Rightarrow}} \sigma B \text{ for some } \sigma \in V^*\}.$$

We then define the Wirth-Weber precedence relations $\doteq$, $\lessdot$, and $\gtrdot$ on $V \times V$:

$A \doteq B$ iff $C \to \sigma AB\tau \in P$ for some $\sigma, \tau \in V^*$;

$A \lessdot B$ iff $C \to \sigma AY\tau \in P$ and $B \in \mathcal{L}_G(Y)$ for some $\sigma, \tau \in V^*$ and $Y \in V_N$;

$A \gtrdot B$ iff either $C \to \sigma XB\tau \in P$ and $A \in \mathcal{R}_G(X)$ for some $\sigma, \tau \in V^*$, $X \in V_N$,

or $C \to \sigma XY\tau \in P$, $A \in \mathcal{R}_G(X)$ and $B \in \mathcal{L}_G(Y)$ for some

$\sigma, \tau \in V^*$, $X, Y \in V_N$.

A grammar for which at most one of the above three relations holds between each pair of symbols is called a Wirth-Weber precedence grammar.

A context-free grammar is called an operator grammar if no production has a right part containing two adjacent non-terminal symbols, i.e. if $P \subseteq V_N \times (V^+ - V^* V_N V_N V^*)$. For any operator grammar $G = (V, V_T, S, P)$, we define the left and right operator terminal sets of a symbol $A \in V_N = V - V_T$:

$$\mathcal{L}_G^o(A) = \{a \in V_T \mid A \overset{+}{\underset{G}{\Rightarrow}} a\tau \text{ or } A \overset{+}{\underset{G}{\Rightarrow}} Ba\tau \text{ for some } \tau \in V^*, B \in V_N\};$$

$$\mathcal{R}_G^o(A) = \{a \in V_T \mid A \overset{+}{\underset{G}{\Rightarrow}} \sigma a \text{ or } A \overset{+}{\underset{G}{\Rightarrow}} \sigma aB \text{ for some } \sigma \in V^*, B \in V_N\}.$$

We then define the operator precedence relations $\doteq^o$, $\lessdot^o$ and $\gtrdot^o$ between each pair of terminal symbols a and b:

$a \doteq^o b$ iff $C \to \sigma ab\tau \in P$ or $C \to \sigma aXb\tau \in P$ for some $\sigma, \tau \in V^*$, $X \in V_N$;

$a \lessdot^o b$ iff $C \to \sigma aY\tau \in P$ and $b \in \mathcal{L}_G^o(Y)$ for some $\sigma, \tau \in V^*$, $Y \in V_N$;

$a \gtrdot^o b$ iff $C \to \sigma Xb\tau \in P$ and $a \in \mathcal{R}_G^o(X)$ for some $\sigma, \tau \in V^*$, $X \in V_N$.

An operator grammar for which at most one of the three operator precedence relations holds between any pair of terminal symbols is called an operator precedence grammar.

A deterministic pushdown store acceptor (or DPDA for short) is a 7-tuple $M = (K, V_T, \Gamma, \delta, q_0, \$, \Gamma_F)$ where k is a finite set of states, V_T is a finite input alphabet, Γ is a finite pushdown store alphabet, $\delta: K \times V_T \times \Gamma \to K \times \{0,1\} \times \Gamma^*$ is the finite control function, $q_0 \in K$ is the initial state, $\$ \in \Gamma$ is the initial pushdown store symbol, and $\Gamma_F \subseteq \Gamma$ is a set of final pushdown store symbols. A configuration is a member of $K \times V_T^* \times \Gamma^*$ which represents the current state of the machine, the remaining input, and the current pushdown store contents. We say that $\langle q, ax, \alpha A\rangle \vdash \langle q', ax, \alpha\beta\rangle$ if $\delta(q,a,A) = \langle q', 0, \beta\rangle$, and $\langle q, ax, \alpha A\rangle \vdash \langle q', x, \alpha\beta\rangle$ if $\delta(q,a,A) = \langle q', 1, \beta\rangle$. If c and c' are configurations, then $c \overset{*}{\vdash} c'$ if there exist configurations $c_0, \ldots, c_k$, $k \geq 0$, such that $c = c_0 \vdash c_1 \vdash \ldots \vdash c_k = c'$. The language accepted by a DPDA, M, is the set

$$L(M) = \{x \in V_T^* \mid \langle q_0, x, \$\rangle \overset{*}{\vdash} \langle q, e, \alpha B\rangle \text{ for some } q \in K, \alpha \in \Gamma^*, \text{ and } B \in \Gamma_F\}.^*$$

Note that we are defining acceptance in terms of final pushdown store symbols instead of final states so that the notion of a DPDA with one state makes sense.

A context-free language is deterministic if it is the language accepted by some DPDA.

3. Backwards Determinism

In this section, we investigate the effects of the backwards-determinism condition on the generative power of precedence grammars. We show that for every operator precedence grammar, we can find an equivalent operator precedence grammar which is backwards-deterministic. However, this theorem fails for Wirth-Weber precedence grammars, for every context-free language has a Wirth-Weber precedence grammar, but only deterministic context-free languages can have BDWWP grammars.

We begin with a technical lemma.

Lemma 3.1. Given any operator precedence grammar, we can find an equivalent operator precedence grammar which has no rule of the form $A \to B$, where B is a single non-terminal symbol.

---

* e is the string of length 0.

Proof: Let $G = (V,V_T,S,P)$ be an operator precedence grammar, and let $V_N = V-V_T$. Let $G' = (V,V_T,S,P')$, where $P' = \{A \to \sigma \mid \sigma \notin V_N$ and for some $B \in V_N$, $A \overset{*}{\underset{G}{\Rightarrow}} B$ and $B \to \sigma \in P\}$. It is readily verified that $L(G') = L(G)$ and that G' is an operator precedence grammar with the desired property. □

Theorem 3.2. Given any operator precedence grammar G, we can find an equivalent backwards-deterministic operator precedence grammar G'. Moreover, the start symbol, S', of G' does not appear on the right side of any production of G', and if $A \to B$ is a production of G' and B is a single nonterminal symbol, then $A=S'$.

Proof: Let $G = (V,V_T,S,P)$ be an operator precedence grammar and let $V_N = V-V_T$. By lemma 3.1, we may assume that G has no rule of the form $A \to B$, $B \in V_N$.

Let $V' = V_T \cup \{X \mid X \subseteq V_N$ and $X \neq \emptyset\} \cup \{S'\}$, S' is a new symbol not in V, and let $V_N' = V'-V_T$. We define a relation "covers" on $(V'-\{S'\})^* \times V^*$ inductively:

i) e covers e;

ii) if $a \in V_T$, then a covers a;

iii) if $X \in V_N'-\{S'\}$ and $A \in X$, then X covers A;

iv) if σ' covers σ and τ' covers τ, then $\sigma'\tau'$ covers $\sigma\tau$.

We now let $G' = (V',V_T,S',P')$, where

$$P' = \{X \to \sigma' \mid X \in V_N'-\{S'\}, \sigma' \in V'^{+} \text{ and } X = \{A \in V_N \mid A \to \sigma \in P \text{ for some } \sigma \in V^{+} \text{ such that } \sigma' \text{ covers } \sigma\}\} \cup \{S' \to X \mid S \in X \in V_N'\}.$$

$L(G') = L(G)$ follows from the two assertions:

(1) For all $X \in V_N'-\{S'\}$, $\sigma' \in V'^*$ and $A \in X$, if $X \overset{*}{\underset{G'}{\Rightarrow}} \sigma'$, then there exists $\sigma \in V^*$ such that σ' covers σ and $A \overset{*}{\underset{G}{\Rightarrow}} \sigma$.

(2) For all $A \in V_N$, $\sigma \in V^*$ and $\sigma' \in V'^*$, if $A \overset{*}{\underset{G}{\Rightarrow}} \sigma$ and σ' covers σ, then there exists an $X \in V_N'-\{S'\}$ such that $A \in X$ and $X \overset{*}{\underset{G'}{\Rightarrow}} \sigma'$.

From assertion (1), it can be verified that for each $X \in V_N'-\{S'\}$, $A \in X$,

$$\mathcal{L}_{G'}^{o}(X) \subseteq \mathcal{L}_{G}^{o}(A)$$

and

$$\mathcal{R}_{G'}^{o}(X) \subseteq \mathcal{R}_{G}^{o}(A).$$

It now follows from the definition of G' that for $a,b \in V_T$, $a \lessdot^{o} b$ in G' implies $a \lessdot^{o} b$ in G; $a \doteq^{o} b$ in G' implies $a \doteq^{o} b$ in G, and $a \gtrdot^{o} b$ in G' implies $a \gtrdot^{o} b$ in G. Hence, any precedence conflict in G' implies a corresponding conflict in G, so G' is operator precedence since G is.

Finally, $X \to Y \in P'$, $Y \in V_N'$ implies $X=S'$ since the right part of every rule of P contains a terminal. Hence, G' is also backwards-deterministic and satisfies the conditions of theorem. □

Corollary 3.3. The class of operator precedence languages is equal to the class of backwards-deterministic operator precedence languages.

For Wirth-Weber precedence grammars, the backwards deterministic restriction does make a difference in the class of languages generated.

Theorem 3.4. Every context-free language has a Wirth-Weber precedence grammar.

Proof: Let L be a context-free language generated by $G = (V,V_T,S,P)$ and let $V_N = V-V_T$. We may assume without loss of generality that every rule of P has one of the forms:

(i) $A \to BC$

or (ii) $A \to a$

where $A,B,C \in V_N$ and $a \in V_T$.[4]

We now define $G' = (V',V_T,S,P')$, where $V' = V_T \cup V_N \cup \hat{V}_N$, $\hat{V}_N = \{\hat{A} \mid A \in V_N\}$, and P' has the productions:

(1) For every rule of the form $A \to BC$ in P, $B,C \in V_N$, P' has the rules $A \to \hat{B}C$ and $\hat{A} \to BC$;

(2) For every rule of the form $A \to a$ in P, $a \in V_T$, P' has the rules $A \to a$ and $\hat{A} \to a$.

Clearly $L(G') = L(G)$. Moreover, G' is Wirth-Weber precedence, for given any $X,Y \in V'$, if any relation exists between them at all, it must be the one given in the table below according to which of the sets V_N, $\hat{V}_N$ or V_T each of the symbols X and Y is in.

	V_N	$\hat{V}_N$	V_T
V_N	$\gtrdot$	$\gtrdot$	$\gtrdot$
$\hat{V}_N$	$\doteq$	$\lessdot$	$\lessdot$
V_T	$\gtrdot$	$\gtrdot$	$\gtrdot$

□

Theorem 3.5. (Wirth and Weber) Every backwards-deterministic Wirth-Weber precedence language is a deterministic context-free language.

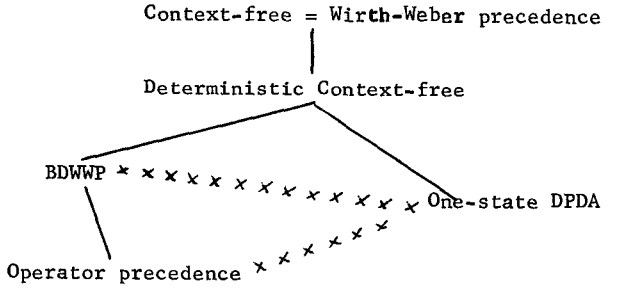
Proof: A flowchart for a deterministic pushdown store machine for recognizing the strings of a BDWWP language is given in Figure 1. We assume the input string is followed by a right endmarker symbol '$\dashv$'. x is a variable in which the current input symbol is stored. $S_0,S_1,S_2,\ldots$ is a pushdown store whose length is k and whose top element is S_k. "Leftpart" is a subroutine which finds the left part of the production whose right side is the argument to the subroutine. If no such production exists, "leftpart" returns with a fail indication. '$\vdash$' is the end-of-stack symbol and is always in S_0. The precedence relations are extended so that for any symbol A of the grammar, $\vdash \lessdot A$ and $A \gtrdot \dashv$. See [2] for a proof that the algorithm works. □

Corollary 3.6. The class of BDWWP languages is properly contained in the class of Wirth-Weber precedence languages.

Proof: There are context-free languages which are not deterministic context-free. □

4. Relations Among Language Classes

In this section, we establish the remaining results summarized in the diagram below. A solid line denotes proper containment and a row of crosses denotes setwise incomparability (i.e. neither set contained in the other).



We begin with a proof that the class of operator precedence languages is contained in the class of BDWWP languages.*

Theorem 4.1. Given an operator precedence grammar, we can find an equivalent BDWWP grammar.

Proof: Let $G = (V,V_T,S,P)$ be an operator precedence grammar and $V_N = V-V_T$. By theorem 3.2, we may assume that G is backwards-deterministic, has no rules of the form $A \to B$, where $B \in V_N$ and $A \neq S$, and S does not appear on the right side of any rule of P.

The new grammar which we construct has each non-terminal symbol of G "paired" with the terminal symbol which appears immediately to its left. The precedence relations between these paired symbols of the new grammar will depend on the precedence relations in G between the terminal symbols which the pairs contain.

There is a slight technical difficulty in that it is possible for non-terminal symbols to appear at the left end of sentential forms and hence have no terminal symbols to their left. Rather than complicate the construction of the Wirth-Weber precedence grammar by treating such symbols as special cases, we introduce a left endmarker symbol '$\vdash$' into the operator precedence grammar, perform the construction of the Wirth-Weber precedence grammar, and finally remove the endmarker from the resulting grammar.

Thus, we let $G' = (V',V_T',S',P')$, where

$V' = V \cup \{S',\vdash\}$, S' and $\vdash$ are new symbols not in V;

$V_T' = V_T \cup \{\vdash\}$;

and

$P' = (P-\{S\to X \mid X \in V_N\}) \cup \{S' \to \vdash X \mid X=S \text{ or } X \in V_N \text{ and } S \to X \in P\}$.

$L(G') = \vdash L(G)$ and G' is a backwards-deterministic operator precedence grammar.

We now define two sets of symbols:

$V_1 = \{<aA> \mid a \in V_T'$, $A \in V_N$, and aA appears in some sentential form of $G'\}$;

$V_2 = \{[\sigma] \mid \sigma \in (V_T'V'^*)$ and $A \to \gamma\sigma$ is in P' for some $A \in V_N'$ and $\gamma \in V'^*\}$.

Let $G'' = (V'',V_T',S',P'')$ where

$V'' = V_T' \cup V_1 \cup V_2 \cup \{S'\}$;

and $P'' = \bigcup_{i=1}^{7} P_i''$, where $P_1'',\ldots,P_7''$ are the following:

$P_1'' = \{<aA> \to <aB>[\sigma] \mid A \to B\sigma \in P'$ and $<aA>$, $<aB> \in V_1$, $[\sigma] \in V_2\}$;

$P_2'' = \{<aA> \to a[\sigma] \mid A \to \sigma \in P'$ and $<aA> \in V_1$, $a \in V_T'$, $[\sigma] \in V_2\}$;

$P_3'' = \{[aA\sigma] \to <aA>[\sigma] \mid [aA\sigma]$, $[\sigma] \in V_2$ and $<aA> \in V_1\}$;

$P_4'' = \{[a\sigma] \to a[\sigma] \mid [a\sigma]$, $[\sigma] \in V_2$ and $a \in V_T'\}$;

$P_5'' = \{[aA] \to <aA> \mid [aA] \in V_2$ and $<aA> \in V_1\}$;

* The author has recently heard that this result was obtained independently by James N. Gray.

$P_6'' = \{[a] \to a \mid [a] \in V_2 \text{ and } a \in V_T'\}$;

$P_7'' = \{S' \to [\vdash A] \mid S' \to \vdash A \in P' \text{ and } [\vdash A] \in V_2\}$.

We first observe that if we "drop" all the brackets of G", we get a context-sensitive grammar which generates exactly L(G'), so clearly $L(G'') \subseteq L(G')$. Now, let

$$\psi: (V_T'V_N \cup V_T')^* \to (V_1 \cup V_T')^*$$

be the function which encloses each member of V_N, together with the terminal symbol immediately to its left, in angle brackets and leaves unchanged terminal symbols which are not immediately followed by a non-terminal symbol, e.g. $\psi(aBccaA) = \langle aB\rangle cc\langle aA\rangle$, where $a,c \in V_T'$, $A,B \in V_N$.

Now, if σ is a sentential form of G' not equal to S' and $\sigma \underset{G'}{\Rightarrow} \tau$, than $\psi(\sigma) \underset{G''}{\overset{*}{\Rightarrow}} \psi(\tau)$ by first applying a rule in $P_1'' \cup P_2''$ and then applying rules of $P_3'' \cup P_4'' \cup P_5'' \cup P_6''$ until all the square brackets have been removed. Hence, $L(G') \subseteq L(G'')$, so we conclude that $L(G'') = L(G')$.

It remains to show that L(G") is a BDWWP grammar. Before we proceed, we state without proof three assertions that are needed to complete the proof.

Assertion 1: If $a \in V_T'$, $A \in V_N$, $b \in \mathcal{L}^0_{G'}(A)$, and aA occurs in a sentential form of G', then $a \lessdot^0 b$ in G'.

Assertion 2: If $[b\tau] \in V_2$ and $X \in \mathcal{L}_{G''}([b\tau])$, then $X=b \in V_T'$ or $X=\langle bB\rangle \in V_1$ for some $B \in V_N$.

Assertion 3: If $\langle bB\rangle \in V_1$, $X=a \in V_T'$ or $X=\langle aA\rangle \in V_1$, and $X \in \mathcal{R}_{G''}(\langle bB\rangle)$, then $a \in \mathcal{R}^0_{G'}(B)$.

To show G" is backwards deterministic, suppose that $X \to \alpha$ and $Y \to \alpha$ are two rules of P" with the same right part α, and suppose that $X \to \alpha$ is in P_i'' and $Y \to \alpha$ is in P_j'', $1 \le i \le j \le 7$. Clearly if i=j, then X=Y since G' is backwards deterministic. If $i \neq j$, then either i=1 and j=3 or i=2 and j=4. Let $\sigma \in V'^*$ be the result of deleting all the brackets in α (i.e. replacing $\langle aA\rangle$ by aA and $[\tau]$ by τ for all $\langle aA\rangle \in V_1$ and $[\tau] \in V_2$). Since i=1 or i=2, $X = \langle aA\rangle \in V_1$, so $\sigma = a\sigma'$ for some σ' such that $A \to \sigma'$ is in P'. Let b be the leftmost terminal symbol of σ'. By assertion 1, $a \lessdot^0 b$ in G'. Since j=3 or j=4, $Y = [\sigma] = [a\sigma']$, so since $Y \in V_2$, there must be some rule $B \to \gamma a\sigma'$ in P'. But then $a \doteq^0 b$, contradicting the fact that G' is an operator precedence grammar. Thus, i=j and X=Y, so G" is backwards deterministic.

To show that G" is Wirth-Weber precedence, we first observe that the only possible precedence relations between symbols of G" (other than S') are those given in the table:

	$V_T' \cup V_1$	V_2
$V_T' \cup V_1$	$\lessdot \ \gtrdot$	$\doteq \ \gtrdot$
V_2	$\gtrdot$	$\gtrdot$

Thus, the only possible precedence conflict between symbols X and Y must occur with $X \in V_T' \cup V_1$ and Y any symbol in V" - {S'}.

The proof is completed by showing (using assertions 2 and 3):

(1) Let $X,Y \in V_T' \cup V_1$ such that X=a or $X=\langle aA\rangle$ and Y=b or $Y=\langle bB\rangle$ for some $a,b \in V_T'$, and $A,B \in V_N$.

a. If $X \lessdot Y$ in G", then $a \lessdot^0 b$ or $a \doteq^0 b$ in G'.

b. If $X \gtrdot Y$ in G", then $a \gtrdot^0 b$ in G'.

(2) Let $X \in V_T' \cup V_1$, $Y = [b\tau] \in V_2$ such that X=a or $X=\langle aA\rangle$ for some $a \in V_T'$ and $A \in V_N$.

a. If $X \doteq Y$ in G", then $a \lessdot^0 b$ or $a \doteq^0 b$ in G'.

b. If $X \gtrdot Y$ in G", then $a \gtrdot^0 b$ in G'.

Any precedence conflict in G" implies a corresponding conflict in G'. Since G' has no conflicts among its operator precedence relations, then G" has no conflicts among its Wirth-Weber precedence relations, so G" is a Wirth-Weber precedence grammar.

We now form the BDWWP grammar for L(G) by deleting the left endmarker ($\vdash$) wherever it occurs in P". By inspection, we see that the modified grammar is indeed backwards deterministic and generates L(G). Such deletion could in general introduce new precedence relations, perhaps causing a precedence conflict, but in this case that does not happen since the deleted symbol could appear only at the left end of sentential forms. Thus, the modified grammar has exactly the same precedence matrix (with the exception of the deleted endmarker) as G" and hence is also a Wirth-Weber precedence grammar. □

We now show that the containment of the class of operator precedence languages in the class of BDWWP languages is proper.

Theorem 4.2. The language

$$L_1 = \{a0^n1^n2^m \mid n,m \ge 1\} \cup \{b0^m1^n2^n \mid n,m \ge 1\}$$

is BDWWP but not operator precedence.

Proof: The following is a BDWWP grammar for L_1:

$$S \to aAX \mid BY$$
$$A \to \hat{0}A1 \mid \hat{0}1$$
$$X \to \hat{2}X \mid \hat{2}$$
$$B \to B0 \mid b0$$
$$Y \to \hat{1}Y2 \mid \hat{1}2$$
$$\hat{0} \to 0$$
$$\hat{1} \to 1$$
$$\hat{2} \to 2.$$

The precedence matrix for this grammar is shown in Figure 2.

To see that L_1 is not operator precedence, one can show directly that in any operator grammar for L_1, $1 \lessdot^{\circ} 1$ and $1 \gtrdot^{\circ} 1$ and hence such a grammar is not precedence. We omit the proof as it is tedious but straightforward. □

Corollary 4.3. The class of operator precedence languages is properly contained in the class of BDWWP languages.

Lemma 4.4. The language

$$L_2 = \{a0^n1^n2^m \mid n,m \geq 1\} \cup \{b0^n1^m2^n \mid n,m \geq 1\}$$

is one-state deterministic but not BDWWP.

Proof: We first exhibit a one-state deterministic pushdown store machine which recognizes L_2.

Let $M = (\{q\}, V_T, \Gamma, \delta, q, \$, \Gamma_F)$, where

$V_T = \{a,b,0,1,2\}$;

$\Gamma = \{\$, A, A', A'', X, F, B, B', B'', Y, G, D\}$;

$\Gamma_F = \{F,G\}$;

and

$\delta(q,a,A) = \langle q, 1, \delta'(A,a)\rangle$ for all $a \in V_T$, $A \in \Gamma$, where for pairs (A,a) in the table below, δ' has the indicated value, and $\delta'(A,a) = D$ (a "dead symbol") for every pair (A,a) not in the table.

(A,a)	$\delta'(A,a)$
$\$,a$	$\$A'$
$A',0$	XA''
$A'',0$	AA''
$A'',1$	e
$A,1$	e
$X,2$	F
$F,2$	F
$\$,b$	$\$B'$
$B',0$	GB''
$B'',0$	BB''
$B'',1$	Y
$Y,1$	Y
$Y,2$	e
$B,2$	e.

To show that L_2 is not BDWWP, we argue that the recognizer M of Figure 1 cannot recognize L_2 and then appeal to the proof of theorem 3.5 to conclude that L_2 is not BDWWP.

Intuitively, any pushdown store recognizer for L_2 must remember whether the first input symbol was an "a" or a "b" and then use its stack (i.e. pushdown store) to remember how many 0's were read so that they may later be compared against the 1's or the 2's, depending on the first symbol. However, the precedence analyzer M has no finite memory in which to remember the first symbol, and it also cannot "pass" information up the stack in the sense that, when the stack grows, the old symbols do not determine which new symbols get placed on the stack. Hence, when M begins reading the 1's, its information about the first input symbol is buried on the stack, so it has no way of "knowing" whether to ignore the 1's and compare its stack with the 2's or whether instead to compare its stack with the 1's, thereby losing the ability to compare later with the 2's.

We now make this argument somewhat more precise, although we omit a complete proof.

Assume M of Figure 1 can recognize L_2. Let Γ be the set of symbols, other than the end-of-stack marker, which can appear on M's stack, and let $\Sigma = \{a,b,0,1,2\}$ be the input alphabet of M (also excluding the right endmarker). A string $\sigma \in \Gamma^*$ is called a stack configuration. We note that if M is at point A in the flowchart of Figure 1 and M's stack $S_1 \ldots S_k = \sigma$, the future behavior of M is completely determined by σ (and of course the future input). Now, if $\sigma \in \Gamma^*$ and $x \in \Sigma^*$, suppose we start M at point A in the algorithm with input x and with σ in the stack. Let $M(\sigma,x)$ denote M's stack configuration after M has read x and has returned again to point A in the algorithm, ready to read the next input.

M has an important property upon which the proof depends. Given a single input symbol, M either pushes that symbol directly on the stack without altering the rest of the stack, or it replaces, perhaps in several steps, some top segment of the stack by a single symbol and then pushes the input symbol on the stack. The choice of the particular symbol with which to replace the top segment of the stack depends only on the part of the stack replaced, not on the remainder of the stack, which remains unchanged, nor on the input. However, the decision of whether to make the replacement or not, and if so, how much of the stack to replace, does depend on both the input and on the remainder of the stack. More precisely, we have:

Lemma: Let $\sigma_1, \sigma_2 \in \Gamma^*$, $\alpha \in \Gamma^+$, and $a \in \Sigma$.
If $\ell(M(\sigma_1\alpha,a)) \geq \ell(\sigma_1) + 2$
and $\ell(M(\sigma_2\alpha,a)) \geq \ell(\sigma_2) + 2$
then there exists $\beta \in \Gamma^+$ such that

$$M(\sigma_1\alpha,a) = \sigma_1\beta a$$

and

$$M(\sigma_2\alpha,a) = \sigma_2\beta a.$$

From this lemma, we can prove that given $A,B \in \Gamma$, and $k \geq 1$, if

$$\ell(M(A0,0^j)) \geq 3$$

and

$$\ell(M(B0,0^j)) \geq 3$$

for all j, $1 \leq j \leq k$, then there exists $\tau_k \in \Gamma^+$ such that

$$M(A0,0^k) = A\tau_k$$

and

$$M(B0,0^k) = B\tau_k.$$

Now, clearly every string in $(a \cup b)\,0^+$ must leave M's stack in a different configuration, for given any strings $x,y \in (a \cup b)0^+$, $x \neq y$, there exists a string $z \in 1^+2^+$ such that $xz \in L_2$ and $yz \notin L_2$. Since there are only finitely many short stack configurations, for all sufficiently large i and j, $\ell(M(e,a0^i)) > 2$ and $\ell(M(e,b0^j)) > 2$.

Let n and m be the greatest integers such that $\ell(M(e,a0^n)) \leq 2$ and $\ell(M(e,b0^m)) \leq 2$. Clearly, $m,n \geq 1$, so in fact $M(e,a0^n) = A0$ and $M(e,b0^m) = B0$ for some $A,B \in \Gamma$. By the above, we conclude then that for all $k \geq 1$, there is some $\tau_k \in \Gamma^*$ such that

$$M(e,a0^{n+k}) = M(A0,0^k) = A\tau_k$$

and $$M(e,b0^{m+k}) = M(B0,0^k) = B\tau_k.$$

Now, consider the behaviors of M with stack $A\tau_k$ and with stack $B\tau_k$ while reading some string in 1^+. Clearly the behaviors must be the same unless the stack eventually gets short. If it does so for infinitely many k, then there must be $k,k' \geq 1$, $k \neq k'$, and $j,j' \geq 1$ such that $b0^{m+k}1^j$ and $b0^{m+k'}1^{j'}$ take M to the same configuration, so M cannot distinguish between

$$b0^{m+k}\,1^j\,2^{m+k} \in L_2$$

and

$$b0^{m+k'}\,1^{j'}\,2^{m+k} \notin L_2.$$

Hence, for all but finitely many $k \geq 1$, the stack cannot get short during the processing of the 1's. But it is then possible to show that for sufficiently large k, there must be $j < n+k$ such that $a0^{n+k}\,1^j\,2$ is accepted by M iff $a0^{n+k}\,1^{n+k}\,2$ is. But the latter is in L_2 while the former is not, contradicting the assumption that M recognizes L_2.

We conclude that L_2 is not a BDWWP language. □

Corollary 4.5. L_2 is not an operator precedence language.

Lemma 4.6. The language

$$L_3 = \{0^n a 1^n a \mid n \geq 1\} \cup \{0^n b 1^n b \mid n \geq 1\}$$

is operator precedence but not one-state deterministic.

Proof: The following is an operator precedence grammar for L_3:

$$S \rightarrow Aa \mid Bb$$
$$A \rightarrow 0A1 \mid 0a1$$
$$B \rightarrow 0B1 \mid 0b1$$

The precedence matrix is:

	a	b	0	1
a				$\doteq^o$
b				$\doteq^o$
0	$\doteq^o$	$\doteq^o$	$<^o$	$\doteq^o$
1	$>^o$	$>^o$		$>^o$

We omit the proof that L_3 is not one-state deterministic. Intuitively, a DPDA for L_3 must build its stack while reading the 0's and pop the stack while reading the 1's, comparing to see that the number of 0's and 1's is the same. But a one-state DPDA, after popping the stack, is in the same configuration whether an "a" or a "b" caused the stack to be popped (although the number of 1's needed to reach that configuration may be different). □

Corollary 4.7. L_3 is a BDWWP language.

Theorem 4.8. The class of one-state deterministic languages is setwise incomparable with the class of BDWWP languages and with the class of operator precedence languages.

Corollary 4.9. The classes of one-state deterministic, BDWWP, and operator precedence languages are all properly contained in the class of deterministic languages.

5. Conclusion

Precedence grammars arise naturally in the study of programming languages. We have answered some of the obvious questions concerning the classes of languages which are definable using the two models of precedence grammars. We have not yet looked at operations on these classes except to note that the language L_2, which was shown not to be BDWWP, is the union of two BDWWP languages, and the language obtained from the BDWWP language L_1 by erasing the first letter of each string is not even deterministic and hence is not BDWWP, so we conclude that the class of BDWWP languages is not closed under union or homomorphism.

References

1. Robert W. Floyd, "Syntactic Analysis and Operator Precedence," JACM 10,3 (July 1963), pp. 316-333.

2. Niklaus Wirth and Helmut Weber, "EULER: A Generalization of ALGOL, and its Formal Definition: Part I," Comm. ACM 9,1 (January 1966), pp. 13-23.

3. Robert McNaughton, "Parenthesis Grammars," JACM 14,3 (July 1967), pp. 490-500.

4. John E. Hopcroft and Jeffrey D. Ullman, Formal Languages and Their Relation to Automata, Addison-Wesley Publishing Company, Reading, Massachusetts, 1969.

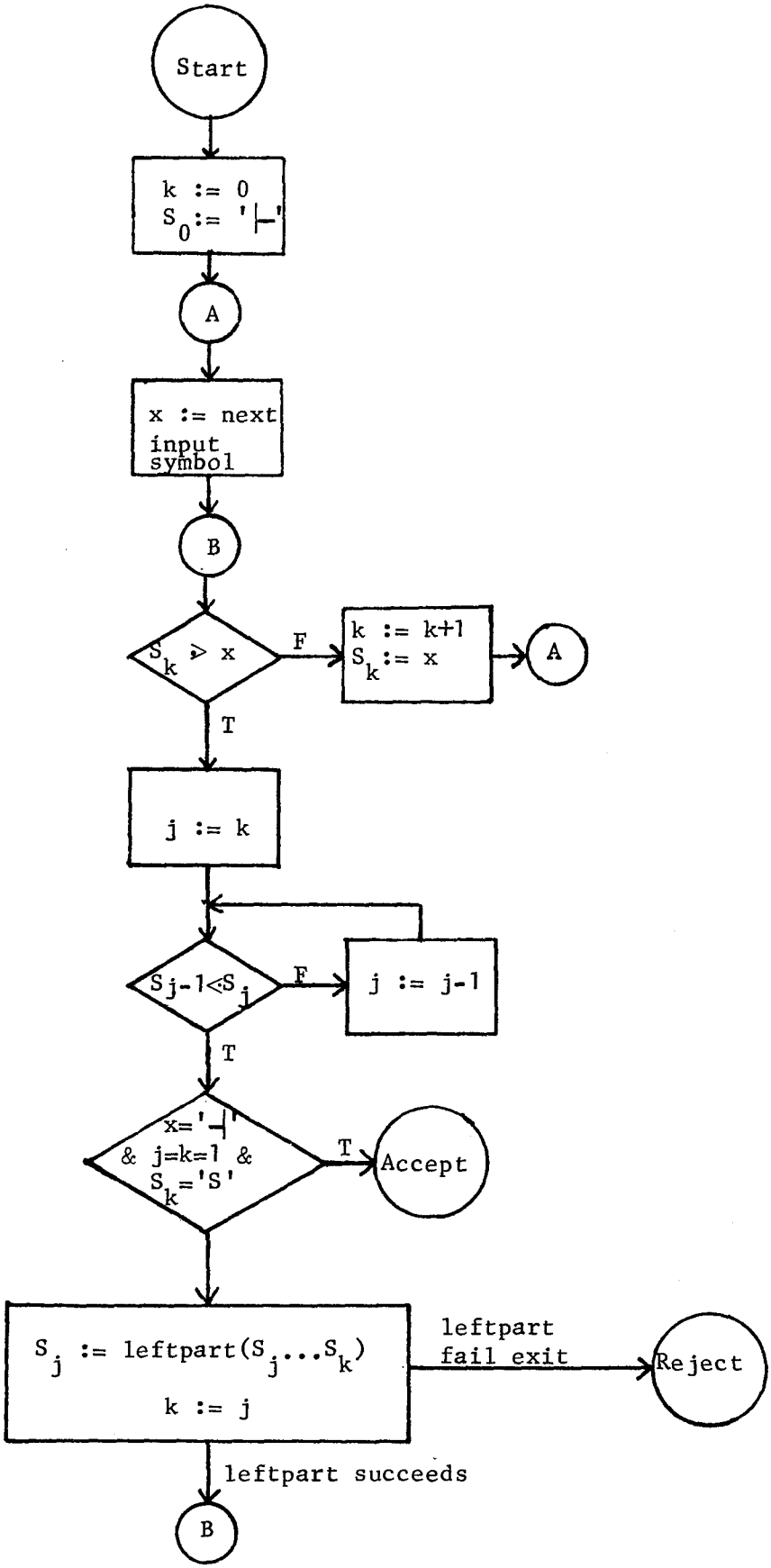


Figure 1. A recognizer for backwards deterministic Wirth-Weber precedence grammars.

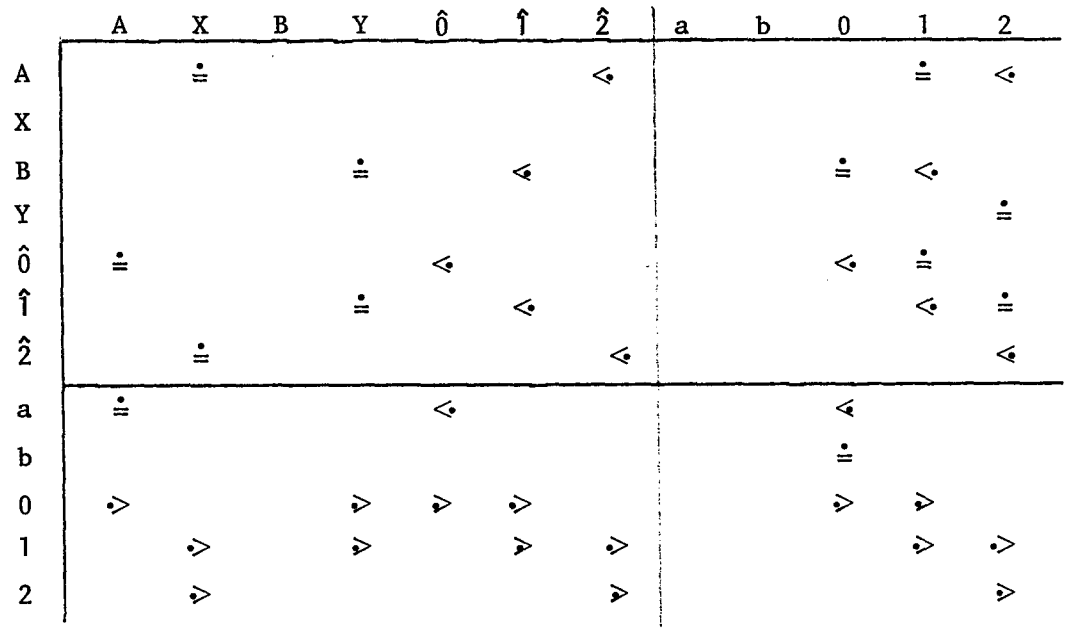


Figure 2. Precedence matrix for the grammar of theorem 4.2.