Algebraic properties of structured context-free languages: old approaches and novel developments

Stefano Crespi Reghizzi[a], Dino Mandrioli[a]

[a]*DEI - Politecnico di Milano Piazza Leonardo da Vinci, 32, I-20133 Milano, Italy*
e-mail: {stefano.crespireghizzi, dino.mandrioli}@polimi.it

Abstract

The historical research line on the algebraic properties of structured CF languages initiated by McNaughton's Parentheses Languages has recently attracted much renewed interest with the Balanced Languages, the Visibly Pushdown Automata languages (VPDA), the Synchronized Languages, and the Height-deterministic ones. Such families preserve to a varying degree the basic algebraic properties of Regular languages: boolean closure, closure under reversal, under concatenation, and Kleene star. We prove that the VPDA family is strictly contained within the Floyd Grammars (FG) family historically known as operator precedence. Languages over the same precedence matrix are known to be closed under boolean operations, and are recognized by a machine whose pop or push operations on the stack are purely determined by terminal characters. We characterize VPDA's as the subclass of FG having a peculiarly structured set of precedence relations, and balanced grammars as a further restricted case. The non-counting invariance property of FG has a direct implication for VPDA too.

1. Introduction

From the very beginning of formal language science, research has struggled with the wish and need to extend as far as possible the nice and powerful properties of regular languages (specifically closure properties). A major initial step has been made by McNaughton with parenthesis grammars [17], characterized by enclosing any righthand side within a pair of parentheses; the alphabet is the disjoint union of internal characters and the pair. By considering instead of strings the stencil or skeletal trees encoded by parenthesized strings, some typical properties of regular languages that do not hold for CF languages are still valid: uniqueness of the minimal grammar, and boolean closure within the class of languages having the same production stencils. Further mathematical developments of such ideas have been pursued in the setting of tree automata [20].

Several decades later, novel motivation arose for the investigation of parentheses-like languages from the interest for mark-up languages such as XML. The *balanced grammars* and languages [2] generalize the parenthesis grammars in two ways: several pairs of parentheses are allowed, and the right-hand side of the grammar rules permit a regular expression over nonterminal and internal symbols to occur between matching parentheses. The property of uniqueness of the minimal grammar is preserved, and the family has the property of closure w.r.t. concatenation and Kleene star, that was missing in parentheses languages. Clearly balanced as well as parentheses languages are closed under reversal.

Model checking and static program analysis provide an entirely different long-standing motivation for such families of languages — those that extend the typical regular properties to infinite-state pushdown systems. To the best of our knowledge the seminal paper of this "new era" is [1] which defines *visibly pushdown*

automata and *languages* (VPDA), a subclass of realtime pushdown automata and deterministic context-free languages. The input alphabet is partitioned into three sets named calls, returns and internals, and the decision of the type of move to perform (push, pop, or a stack neutral move) is determined by the membership of the current input character; in other words the type of a move is solely input-driven. VPDA languages extend balanced grammars in two ways that are important for modelling symbolic program execution: they allow parentheses to remain unbalanced to represent an execution state where some procedures have not returned, and a call symbol can be matched by two or more return symbols to represent procedures with multiple exits. For each partitioned alphabet the corresponding language family is closed under the regular operations, including complement. VPDA's can be determinized and reversal produces a VPDA with calls and returns interchanged. We observe that the intended applications to static program analysis need closure under reversal in order to compute the pre- and post-reachability sets.
Impulsed by this new approach, a variety of extensions and specializations of the original class have been proposed and investigated. Among them, we mention the following. The *synchronized pushdown automata* [3], instead of the fixed 3-partition of VPDA's, use a finite transducer that determines the type of move the PDA must perform.
The *height-deterministic automata* [18] further extended the previous idea by considering the class of PDA's characterized by the same integer-valued function returning the height of the stack for each input string; within this approach the deterministic and the real-time cases are singled out for having richer closure properties. Last, the *synchronized grammars* [4] are a more comprehensive model that uses an input-driven pushdown transducer to decide the type of a move. Not surprisingly, such more general models lose certain nice properties of VPL, in particular the closure under reversal, concatenation, and Kleene star.
Short after McNaughton's results, we investigated similar closure properties of *Floyd's* operator precedence *Grammars* [12] [1] (FG), an elegant precursor of LR(k) grammars, also exploited by one of us in his work on grammar inference [6]. For any given precedence matrix a syntax tree stencil is defined a priori for any word that is generated by any FG having the same precedence matrix. The family of such Floyd grammars and the related languages are a boolean algebra [9]. We also extended the notion of non-counting regular language of McNaughton and Papert [19] to the parentheses languages [7] and to FG [8].
In this paper we resume the study of FG in the perspective of the cited grammatical models. We show that VPDA is a special case of FG characterized by a very restricted structure of the precedence relations, thus providing a new characterization of VPDA in terms of operator grammars. Further restrictions are shown for the case of balanced languages. Then we compare FG with the height-deterministic family showing strict inclusion, and that reversal closure is lost by that generalization.
The paper is structured as follows: Section 2 provides the essential definitions of the main classes of languages (defined through automata and/or grammars) that will be considered in this paper (others will be referred only on the basis of previous literature); Section 3 investigates the mutual inclusion relations among them. Section 4 compares the same classes of languages w.r.t their closure properties. The conclusion mentions that the non-counting invariance property of FG has a direct implication for VPDA too and shows that the whole picture of such language families deserves further analysis to answer a few remaining open issues.

[1] We propose to name them *Floyd grammars* to honor the memory of Robert Floyd and also to avoid confusion with other similarly named but quite different types of precedence grammars.

2. Basic definitions

We list the essential definitions of parentheses and balanced grammars, VPDA, height-deterministic automata, and Floyd grammars. For brevity, other classes are not defined here because they can be somewhat put in relation with the above "basic" ones. They are nevertheless taken into consideration in Section 4. The same name is given to a class of devices (grammars or automata) and to the class of languages that can be defined by means of them.

The empty string is ε, the terminal alphabet is Σ. For a string x and a letter a, $|x|_a$ denotes the number of occurrences of letter a, and extend the notation to $|x|_\Delta$, for a set $\Delta \subseteq \Sigma$. Let $first(x)$ and $last(x)$ denote the first and last letter of $x \neq \varepsilon$. The projection of a string $x \in \Sigma^*$ on Δ is denoted $\pi_\Delta(x)$.

The operators union, concatenation, and Kleene star are called *regular*. A *regular expression* is a formula written using the regular operators, parentheses and characters from a specified alphabet.

A *Context-Free* CF grammar is a 4-tuple $G = (V_N, \Sigma, P, S)$, where V_N is the nonterminal alphabet, P the production set, and S the axiom. An *empty rule* has ε as the right part. A *renaming rule* has one nonterminal as right part. A grammar is *invertible* if no two productions have identical right parts.

A production has the *operator form* if its right part has no adjacent nonterminals, and an *operator grammar* (OG) contains just such productions. Any CF grammar admits an equivalent OG, which can be also assumed to be invertible [14].

For a CF grammar G over Σ, the associated *parenthesis grammar* [17] $\widetilde{G}$ has the rules obtained by enclosing each right part of a rule of G within the parentheses '[' and ']' that are assumed not to be in Σ.

A *balanced grammar* [2] is a CF grammar has a terminal alphabet partitioned into $\Sigma = \Sigma_{par} \cup \Sigma_i$, where $\Sigma_{par} = \{a, \overline{a}, b, \overline{b}, \ldots\}$ is a set of *matching parentheses* and the elements of Σ_i are named *internal*. Let V_N be the nonterminal alphabet. Every rule of a balanced grammar has the form $X \to a\alpha\overline{a}$ or $X \to \alpha$, where α is a regular expression over $V_N \cup \Sigma_i$. The corresponding family is denoted BALAN.

A *pushdown automaton* PDA $\mathcal{A}$ over an alphabet Σ is a tuple $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where the initial state $q_0 \in Q$ and $F \subseteq Q$ are the final states. Γ is the stack alphabet containing $\perp$, the stack bottom symbol. The transition relation is

$\delta \subseteq Q \times \Gamma \times (\Sigma \cup \varepsilon) \times Q \times (\Gamma \setminus \{\perp\})^*$

The notation [18] $pX \xrightarrow{a} q\alpha$ is equivalent to $(p, X, a, q, \alpha) \in \delta$.

A PDA is called *realtime* (RPDA) if $pX \xrightarrow{a} q\alpha$ implies $a \neq \varepsilon$.

A PDA is called *deterministic* (DPDA) if for every $p \in Q, X \in \Gamma$ and $a \in \Sigma \cup \{\varepsilon\}$ we have $|\{q\alpha \mid pX \xrightarrow{a} q\alpha\}| \leq 1$ and if $pX \xrightarrow{\varepsilon} q\alpha$ and $pX \xrightarrow{a} q'\alpha'$ then $a = \varepsilon$

A *realtime deterministic* automaton is denoted RDPDA.

The set $Q\Gamma^*$ is the set of *configurations* of a PDA, with *initial* configuration $q_0\perp$.

The *labelled transition system* generated by $\mathcal{A}$ is the edge-labeled directed graph

$$\left(Q\Gamma^*\perp \,,\, \bigcup_{a \in \Sigma \,\cup\, \{\varepsilon\}} \xrightarrow{a} \right)$$

Given a string $w \in \Sigma^*$, we write $p\alpha \stackrel{w}{\Longrightarrow} q\beta$ if there exists a finite w'-labelled path, $w' \in (\Sigma \cup \{\varepsilon\})^*$, from $p\alpha$ to $q\beta$, and w is the projection of w' onto Σ. Notice that according to [18] the w'-labelled path includes transitions of the type $\xrightarrow{\varepsilon}$.

An $\mathcal{A}$ is *complete* if $\forall w \in \Sigma^*,\ q_0\perp \stackrel{w}{\Longrightarrow} q\alpha$.

The language *recognized* by $\mathcal{A}$ is $L(\mathcal{A}) = \{w \in \Sigma^* \mid q_0\perp \stackrel{w}{\Longrightarrow} p\alpha,\ p \in F\}$

A PDA $\mathcal{A}$ is *normalized* [18] if

1. $\mathcal{A}$ is complete;

2. for all $p \in Q$, all rules in δ of the form $pX \xrightarrow{a} q\alpha$ either satisfy $a \in \Sigma$, or all of them satisfy $a = \varepsilon$, but not both;
3. every rule in δ is of the form
   - $pX \xrightarrow{a} q$
   - $pX \xrightarrow{a} qX$
   - $pX \xrightarrow{a} qYX$ where $a \in \Sigma \cup \{\varepsilon\}$

For a normalized PDA moves are named *push* if $|\alpha| = 2$, *pop* if $|\alpha| = 0$, and internal if $|\alpha| = 1$. The normalization preserves the characteristics of DPDA, RPDA and RDPDA devices.

Height-determinism

Let $w \in (\Sigma \cup \{\varepsilon\})^*$. The set $N(\mathcal{A}, w)$ of *stack heights* reached by $\mathcal{A}$ after reading w is $\{|\alpha| \mid q_0\bot \overset{w}{\Longrightarrow} q\alpha\bot\}$. A *height-deterministic* PDA (HPDA) is a PDA that is normalized and such that $|N(A, w)| \leq 1$ for every $w \in (\Sigma \cup \{\varepsilon\})^*$.
The families of height-deterministic PDA's, DPDA's, and RDPDA's (and languages) are resp. denoted by HPDA, HDPDA, and HRDPDA.
A normalized DPDA is an HDPDA and the language families HPDA and CF coincide [18].
Two HPDA's $\mathcal{A}_1$ and $\mathcal{A}_2$ over the same alphabet Σ are in the equivalence relation *H-synchronized*, denoted by $\mathcal{A}_1 \sim_H \mathcal{A}_2$, if $N(\mathcal{A}_1, w) = N(\mathcal{A}_2, w)$ for every $w \in (\Sigma \cup \{\varepsilon\})^*$.
Let $[\mathcal{A}]_{\sim_H}$ denote the equivalence class containing the HPDA $\mathcal{A}$ and $\mathcal{A} - HPDA$ denote the class of languages recognized by any HPDA H-synchronized with $\mathcal{A}$.

Visibly pushdown automata

A *visibly pushdown* (VP) [1] alphabet is a 3-tuple $\widehat{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_i \rangle$, with Σ the disjoint union of the three sets. Elements of the three sets are resp. termed *calls*, *returns* and *internal* letters. A *VP automaton* VPDA is a PDA $\mathcal{A} = (\Sigma, Q, q_0, \Gamma, \delta, F)$, where $\widehat{\Sigma}$ is a VP alphabet. The transition relation is

$$\delta \subseteq (Q \times \Sigma_c \times Q \times (\Gamma \setminus \{\bot\}) \quad \cup \quad (Q \times \Sigma_r \times \Gamma \times Q) \quad \cup \quad (Q \times \Sigma_i \times Q)$$

that can be readily seen to specialize the previous definition for a general PDA.

Floyd grammars

The definitions for operator precedence grammars, here renamed *Floyd Grammars* (FG), are from [9]. (See [13] for a recent presentation.)
For a nonterminal A of an OG G, the *left and right terminal sets* are

$$\mathcal{L}_G(A) = \{a \in \Sigma \mid A \overset{*}{\Rightarrow} Ba\alpha\} \qquad \mathcal{R}_G(A) = \{a \in \Sigma \mid A \overset{*}{\Rightarrow} \alpha aB\}$$

where $B \in V_N \cup \{\varepsilon\}$ and $\overset{*}{\Rightarrow}$ denotes, as usual, a derivation. The two definitions are extended to a set W of nonterminals and to a string $\beta \in V^+$ via

$$\mathcal{L}_G(W) = \bigcup_{A \in W} \mathcal{L}_G(A) \text{ and } \mathcal{L}_G(\beta) = \mathcal{L}_{G'}(D)$$

where D is a new nonterminal and G' is the same as G except for the addition of the production $D \to \beta$. Finally $\mathcal{L}_G(\epsilon) = \emptyset$. The definitions for $\mathcal{R}$ are similar.
For an OG G, let $\alpha, \beta \in (V_N \cup \Sigma)^*$ and $a, b \in \Sigma$, three binary operator precedence (OP) relations are defined:

equal precedence:	$a \doteq b$ iff $\exists A \rightarrow \alpha a B b \beta$, $B \in V_N \cup \{\varepsilon\}$;
yields precedence:	$a \dot{>} b$ iff $\exists A \rightarrow \alpha D b \beta$, $D \in V_N$ and $a \in \mathcal{R}_G(D)$
takes precedence:	$a \dot{<} b$ iff $\exists A \rightarrow \alpha a D \beta$, $D \in V_N$ and $b \in \mathcal{L}_G(D)$;

For an OG G, the *operator precedence matrix* (OPM) $M = OPM(G)$ is a $|\Sigma| \times |\Sigma|$ array that to each ordered pair (a, b) associates the set M_{ab} of OP relations holding between a and b. Given two OPM's M_1 and M_2, we define
$M_1 \subseteq M_2 \iff M_{1,ab} \subseteq M_{2,ab}, \quad M = M_1 \cup M_2 \iff M_{ab} = M_{1,ab} \cup M_{2,ab}; \forall a, b.$
G is a *Floyd grammar* FG if, and only if, $OPM(G)$ is a *conflict-free* matrix, i.e., $\forall a, b$, $|OPM(G)_{ab}| \leq 1$.
Two matrices are *compatible* if their union is conflict-free.
A FG is in *Fischer normal form [10]* if it is invertible, the axiom S does not occur in the right part of any production, and there are no renaming productions, except those with left part S (if any).

For the reader convenience the acronyms are collected in the table:

BALAN	balanced grammar
CF	context-free
DPDA	deterministic pushdown automaton
FG	Floyd grammar
HDPDA	height-deterministic deterministic pushdown automaton
HPDA	height-deterministic pushdown automaton
HRDPDA	height-deterministic realtime deterministic pushdown automaton
OG	operator grammar
OPM	operator precedence matrix
REG	regular language
RDPDA	realtime deterministic pushdown automaton
RPDA	realtime pushdown automaton
PDA	pushdown automaton
VPDA	visibly pushdown automaton

3. Containment relations

First we recall some of the relevant known [1, 18, 16, 4] containment relations between some recent language families, then we position FG within the picture. The main strict inclusions are:

$$REG \subset BALAN \subset VPDA \subset HRDPDA = RDPDA \subset HDPDA = DPDA$$

Notice that the above inclusions preserve the structural properties of the languages: for instance if the partition of a VP alphabet places a letter in Σ_c and therefore associates a push move to it, the corresponding HDPDA automaton too performs a push move on that letter.
The first [3] and second [5] family of Caucal, as well as the one of Fisman and Pnueli [11] fall in between VPDA and DPDA. but lack of space prevents a detailed presentation.
Next we focus on FG languages. It is well-known that $FG \subset DPDA$. On the other hand, FG includes non-realtime deterministic languages such as $L_1 = \{a^m b^n c^n d^m \mid m, n \geq 1\} \cup \{a^m b^+ e d^m \mid m \geq 1\}$. Observing that $L_2 = \{a^n c a^n \mid n \geq 0\}$ is in HRDPDA but not in FG, since, by an elementary application of the pumping lemma, this would imply a precedence conflict, we have:

Proposition 3.1. *The families of FG and HRDPDA languages are incomparable.*

Our main result is that the VPDA languages are a well-characterized special case of FG languages. First we give a construction from a VPDA to a FG having a certain type of precedence matrix, second we construct a VPDA for any FG with such matrices. At last we include also BALAN in the matrix-based characterization. We need to analyze the structure of VPDA strings. A string in $\{c, r\}^*$ is *well parenthesized* if it reduces to ε via the cancellation rule $cr \to \varepsilon$.
Let ρ be the alphabetical mapping from $\Sigma_c \cup \Sigma_r \cup \Sigma_i$ to $\{c, r\}$ defined by $\rho(c_j) = c, \forall c_j \in \Sigma_c$, $\rho(r_j) = r, \forall r_j \in \Sigma_r$, and $\rho(s_j) = \varepsilon, \forall s_j \in \Sigma_i$. A non-empty string $x \in \Sigma^*$ is *well balanced* if $\rho(y)$ is well parenthesized; it is *well closed* if in addition $first(x) \in \Sigma_c$ and $last(x) \in \Sigma_r$.
Let $\mathcal{A} = (Q, \Sigma, Q, q_0, \Gamma, \delta, Q_F)$ be a VPDA, with $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$.

Lemma 3.2. *Any string $x \in L(\mathcal{A})$ can be factorized as*
$x = yc_0z$ or $x = y$, with $c_0 \in \Sigma_c$, such that

1. *$y = u_1w_1u_2w_2 \ldots u_kw_k, k \geq 0$, where $u_j \in (\Sigma_i \cup \Sigma_r)^*$, and $w_j \in \Sigma^*$ is a, possibly missing, well-closed string;*
2. *$z = v_1c_1v_2c_2 \ldots c_{r-1}v_r, \quad r \geq 0$, where $c_j \in \Sigma_c$ and $v_j \in \Sigma^*$ is a, possibly null, well-balanced string.*

Proof *Let the transitions from state q to q' be labelled as follows: $(r, \perp)$ denotes a move of type $(q, r, \perp, q') \in \delta_r$; (r, Z) denotes a move of type $(q, r, Z, q') \in \delta_r$ with $Z \neq \perp$; $\frac{c}{Z}$ denotes a move of type $(q, c, q', Z) \in \delta_c$; s denotes a move of type $(q, s, q') \in \delta_s$.*
We examine the possible sequences of moves of a suitable VPDA $\mathcal{A}$ that for convenience is non-deterministic (determinization is always possible [1]). We only discuss the case $x = yc_0z$, since the case $x = y$ is simpler. The computation starts with a series of moves in $\{(r, \perp) \mid s\}^*$, which scan the prefix u_1 and leave the stack empty.
Then the machine may do a series of moves to scan string w_1. The first move is of type $\frac{c}{Z_i}$. The move is possibly followed by a nested computation scanning a well-balanced string, and at last by a move of type (r, Z_i). The effect is to scan a well balanced string w_1. Clearly the nested computation may also include internal moves.
After scanning w_1 the stack is empty, and the computation may scan u_2, and so on, until w_k is scanned.
Alternatively and non-deterministically, when the stack is empty, the machine may perform a move $\frac{c_0}{Z_U}$, thus entering the phase that scans string z. We denote as Z_U a symbol written on the stack, which will never be touched by a subsequent pop move. In other words, c_0 is nondeterministically assumed to be an unmatched call.
Then the z phase non-deterministically scans a well balanced string v_1. Then, again nondeterministically, it may perform a move $\frac{c_1}{Z_U}$. Then it may scan another well balanced string v_2, and so on, ending with a stack in $\perp {Z_U}^+$.
At any time, when the machine enters a final state, it may halt and recognize the scanned input.

Clearly string y is the longest prefix such that the accepting computation ends with empty stack. For simplicity, without loss of generality, we assume that no transition enters the initial state q_0. For convenience we shall denote by a subscripted letter q the states traversed while scanning y, and by a subscripted letter p the states traversed in the computation of c_0z. The state set is thus partitioned into $Q = \{q_0\} \cup Q_q \cup Q_p$.

Since VPL's are CF languages, previous papers (e.g. [21]) have also used grammars to define them, but such grammars are not OG or have precedence conflicts; instead, we present a construction producing a grammar with the required properties.

Theorem 3.3. *For any visibly pushdown automaton $\mathcal{A}$ a Floyd grammar G such that $L(G) = L(\mathcal{A})$ can be effectively constructed.*

Proof *First we construct the grammar, then we prove that it is an FG, and lastly that it is equivalent to $\mathcal{A}$.*

Grammar construction.
The productions are keyed to the factorization of Lemma 3.2 and are listed in Tables 1, 2, 3, and 4. The scheme of a sample syntax tree produced by the grammar, for a string factorized as in Lemma 3.2, is shown in Fig. 1.

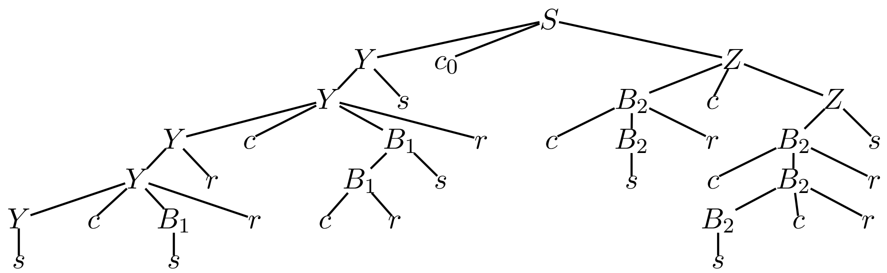

Figure 1: Schema of a syntax tree generated by the precedence grammar constructed in Theor. 3.3.

	Σ_c	Σ_r	Σ_i
Σ_c	$\lessdot$	$\doteq$	$\lessdot$
Σ_r	$\gtrdot$	$\gtrdot$	$\gtrdot$
Σ_i	$\gtrdot$	$\gtrdot$	$\gtrdot$

Figure 2: *Total VP precedence matrix M_T.*

Nonterminals of class Y generate a string such that the automaton, parsing it, starts and ends with empty stack. Nonterminals of classes B_1, B_2 derive a well-balanced (but not necessarily well-closed) string. Nonterminals of class Z derive a string such that, starting with a non-empty stack of the form $\perp Z_U^+$, the stack never pops a Z_U and at last contains a string in $\perp Z_U^+$.
The nonterminal symbols of the grammar are denoted by a pair of states $\langle q_i, q_j \rangle$ or $\langle p_i, p_j \rangle$, or by a triple $\langle q_i, Z, q_j \rangle$ or $\langle p_i, Z, p_j \rangle$, with $Z \in \Gamma$. Intuitively, a nonterminal of the generic form $\langle r_i \dots r_j \rangle$ generates a terminal string u if, and only if, there is a computation of the machine from the left state r_i to the right state r_j which reads the same string and never modifies the initial stack. Furthermore, nonterminals $\langle q_i, q_j \rangle$ leave the stack unchanged; nonterminals $\langle p_i, p_j \rangle$ at most increase the number of Z_U's; and nonterminals $\langle q_i, Z, q_j \rangle$ or $\langle p_i, Z, p_j \rangle$ denote that the computation starts and ends with Z on the top and generates a well-balanced terminal string w.
To construct the productions we examine the transitions of the VPDA. In what follows, calls, returns and internal characters are respectively denoted c, r and s; Z, W are stack symbols different from $\perp$. Notice that the grammar constructed may be not reduced (i.e. some nonterminal may be unreachable from the axiom or it may not derive any terminal string). In that case the useless nonterminals and productions can be removed by well-known algorithms (e.g. in [15]).

Table 1: Productions of the axiom.

case	transitions	productions
$S \to Yc_0Z$	$\delta(q_i, c_0) \ni (p_j, Z_U)$	$S \to \langle q_0, q_i\rangle c_0\langle p_j, p_f\rangle, \forall p_f \in F$
$S \to Y$		$S \to \langle q_0, q_f\rangle, \forall q_f \in F$
$S \to Yc_0$	$\delta(q_i, c_0) \ni (p_f, Z_U), p_f \in F$	$S \to \langle q_0, q_i\rangle c_0$
$S \to c_0Z$	$\delta(q_0, c_0) \ni (p_j, Z_U)$	$S \to c_0\langle p_j, p_f\rangle, \forall p_f \in F$
$S \to c_0$	$\delta(q_0, c_0) \ni (p_f, Z_U), p_f \in F$	$S \to c_0$

Table 2: Productions of nonterminals of class Y (deriving the maximal prefix ending with empty stack).

case	transitions	productions
$Y \to s$	$\delta(q_0, s) \ni q_i$	$\langle q_0, q_i\rangle \to s$
$Y \to r$	$\delta(q_0, r, \perp) \ni q_i$	$\langle q_0, q_i\rangle \to r$
$Y \to Ys$	$\delta(q_i, s) \ni q_j$	$\langle q_0, q_j\rangle \to \langle q_0, q_i\rangle s$
$Y \to Yr$	$\delta(q_i, r, \perp) \ni q_j$	$\langle q_0, q_j\rangle \to \langle q_0, q_i\rangle r$
$Y \to cBr$	$\delta(q_0, c) \ni (q_t, Z)$ and $\delta(q_k, r, Z) \ni q_h$	$\langle q_0, q_h\rangle \to c\langle q_t, Z, q_k\rangle r$
$Y \to cr$	$\delta(q_0, c) \ni (q_t, Z)$ and $\delta(q_t, r, Z) \ni q_h$	$\langle q_0, q_h\rangle \to cr$
$Y \to YcBr$	$\delta(q_i, c) \ni (q_j, Z)$ and $\delta(q_m, r, Z) \ni q_n$	$\langle q_0, q_n\rangle \to \langle q_0, q_i\rangle c\langle q_j, Z, q_m\rangle r$
$Y \to Ycr$	$\delta(q_i, c) \ni (q_j, Z)$ and $\delta(q_m, r, Z) \ni q_n$ and $q_j = q_m$	$\langle q_0, q_n\rangle \to \langle q_0, q_i\rangle cr$

Table 3: Productions for nonterminals of classes B_1 and B_2, generating well-balanced string. (The case B_2 just differs with respect to the state set, which is Q_p instead of Q_q.)

case	transitions	productions
$B \to BcBr$	$\delta(q_i, c) \ni (q_j, Z)$ and $\delta(q_m, r, Z) \ni q_n$	$\langle q, q_n\rangle \to \langle q, q_i\rangle c\langle q_j, Z, q_m\rangle r$, $\forall q \in Q_q$
$B \to Bcr$	$\delta(q_i, c) \ni (q_j, Z)$ and $\delta(q_j, r, Z) \ni q_n$	$\langle q, q_n\rangle \to \langle q, q_i\rangle cr$, $\forall q \in Q_q$
$B \to cBr$	$\delta(q_i, c) \ni (q_j, Z)$ and $\delta(q_m, r, Z) \ni q_n$	$\langle q_i, q_n\rangle \to c\langle q_j, Z, q_n\rangle r$
$B \to cr$	$\delta(q_i, c) \ni (q_j, Z)$ and $\delta(q_j, r, Z) \ni q_n$	$\langle q_i, q_n\rangle \to cr$, $\forall q \in Q_q$
$B \to BcBr$	$\delta(q_i, c) \ni (q_j, Z)$ and $\delta(q_m, r, Z) \ni q_n$	$\langle q_i, W, q_n\rangle \to \langle q, q_i\rangle c\langle q_j, Z, q_m\rangle r$, $\forall q \in Q_q, W \in \Gamma$
$B \to cBr$	$\delta(q_i, c) \ni (q_j, Z)$ and $\delta(q_m, r, Z) \ni q_n$	$\langle q, W, q_n\rangle \to c\langle q_j, Z, q_m\rangle r, \forall q \in Q_q, W \in \Gamma$
$B \to Bcr$	$\delta(q_i, c) \ni (q_j, Z)$ and $\delta(q_j, r, Z) \ni q_n$	$\langle q, W, q_n\rangle \to \langle q, q_i\rangle cr$, $\forall q \in Q_q, W \in \Gamma$
$B \to Bs$	$\delta(q_h, s) \ni q_m$	$\langle q, W, q_m\rangle \to \langle q, q_h\rangle s$, $\forall q \in Q_q, W \in \Gamma$
$B \to s$	$\delta(q_j, s) \ni q_m$	$\langle q_j, Z, q_m\rangle \to s$, $\forall W \in \Gamma$

G is a Floyd grammar

By construction all the productions are in operator form. To verify that the operator precedence matrix M is conflict-free, it suffices to compute the relevant terminal sets the matrix entries using the previous

Table 4: Productions for nonterminals of class Z.

case	transitions	productions
$Z \to cZ$	$\delta(p_i, c) \ni (p_j, Z_U)$	$\langle p_i, p_f\rangle \to\ c\langle p_j, p_f\rangle, \forall p_f \in F$
$Z \to c$	$\delta(p_i, c) \ni (p_f, Z_U), p_f \in F$	$\langle p_i, p_f\rangle \to c$
$Z \to BcZ$	$\delta(p_j, c) \ni (p_h, Z_U)$	$\langle p, p_f\rangle \to \langle p, p_j\rangle c\langle p_h, p_f\rangle, \forall p_f \in F, p \in Q_p$
$Z \to BC$	$\delta(p_j, c) \ni (p_f, Z_U), p_f \in F$	$\langle p, p_f\rangle \to \langle p, p_j\rangle c$
$Z \to BcBr$	$\delta(p_i, c) \ni (p_j, Z)$ and $\delta(p_m, r, Z) \ni p_n$	$\langle p, p_f\rangle \to \langle p, p_i\rangle c\langle p_j, Z, p_m\rangle r$, $\forall p \in Q_p, p_f \in F$
$Z \to cBr$	$\delta(p_i, c) \ni (p_j, Z)$ and $\delta(p_m, r, Z) \ni p_n$	$\langle p_i, p_f\rangle \to c\langle p_j, Z, p_n\rangle r, \forall p_f \in F$
$Z \to Bcr$	$\delta(p_i, c) \ni (p_j, Z)$ and $\delta(p_j, r, Z) \ni p_n$	$\langle p, p_f\rangle \to \langle p, p_i\rangle cr, \forall p \in Q_p\ , \forall p_f \in F$
$Z \to cr$	$\delta(p_i, c) \ni (p_j, Z)$ and $\delta(p_j, r, Z) \ni p_n$	$\langle p_i, p_f\rangle \to cr, \forall p \in Q_p, p_f \in F$
$Z \to Bs$	$\delta(p_j, s) \ni p_f, p_f \in F$	$\langle p, p_f\rangle \to \langle p, p_j\rangle s, \forall p \in Q_p$
$Z \to s$	$\delta(p_j, s) \ni p_f\ , p_f \in F$	$\langle p, p_f\rangle \to s$

definitions. It should be enough to show one case.
For the production $\langle q_0, q_n\rangle \to \langle q_0, q_i\rangle c\langle q_j, Z, q_m\rangle r$ the set $\mathcal{R}_G(\langle q_0, q_i\rangle) \subseteq \Sigma_i \cup \Sigma_r$ produces the relations $s\dot{>}c, r\dot{>}c$. The sets $\mathcal{L}_G(\langle q_j, Z, q_m\rangle) \subseteq \Sigma_i \cup \Sigma_c$, $\mathcal{R}_G(\langle q_j, Z, q_m\rangle) \subseteq \Sigma_i \cup \Sigma_r$ determine $c\dot{<}c, c\dot{<}s$ and $s\dot{>}r, r\dot{>}r$; the right part of the production gives $c\doteq r$. Thus we obtain a conflict-free matrix $M \subseteq M_T$ where M_T is the total matrix in Fig. 2.

Fig. 3 reproduces the string of Fig. 1 with precedence relations between characters that are consecutive or separated by a nonterminal.

Proof that $L(G) = L(\mathcal{A})$

It is obtained by a fairly natural induction showing the double implication between computations and derivations. It is structured into several "macro-steps" mirroring the factorization introduced in Lemma 3.2. We develop in detail only a sample of the various cases, since the others are similar.

1. $(q_i, \perp) \overset{*}{\underset{x}{\mapsto}} (q_j, \perp) \iff \langle q_i, q_j\rangle \overset{*}{\Rightarrow} x, x \in (\Sigma_r \cup \Sigma_i)^*$.

2. $(q_i, \sigma) \overset{*}{\underset{x}{\mapsto}} (q_j, \sigma) \iff \langle q_i, q_j\rangle \overset{*}{\Rightarrow} x, x \in \Sigma^*$ and well-balanced.

3. $(p_i, \sigma) \overset{*}{\underset{x}{\mapsto}} (p_j, \sigma) \iff \langle p_i, p_j\rangle \overset{*}{\Rightarrow} x, x \in \Sigma^*$ and well-balanced.

4. $(p_i, \perp Z_U^k) \overset{*}{\underset{c^n}{\mapsto}} (p_j, \perp Z_U^{k+n}) \iff \langle p_i, p_j\rangle \overset{*}{\Rightarrow} c^n$.

5. $\forall \gamma \in \Gamma^*, Z, (p_i, \perp\gamma Z) \overset{*}{\underset{w}{\mapsto}} (p_j, \perp\gamma Z)$ (without ever popping Z) $\iff \langle p_i, Z, p_j\rangle \overset{*}{\Rightarrow} w$, where w is a well-balanced string.
   Induction base:
   (a) $\delta(p_i, c) \ni (p_k, Z) \wedge \delta(p_k, r, Z) \ni p_r \iff \exists W : \langle p_i, W, p_j\rangle \to cr$
   (b) $\delta(p_i, s) \ni p_j \iff \exists W : \langle p_i, W, p_j\rangle \to s$
   From the inductive hypotheses:

$$\vdash - \dashv$$

$$\vdash \lessdot - - - - - - - - - - - c \gtrdot - \dashv$$

$$\vdash \lessdot - - - - - - - - - - - c \lessdot - - - - - - - - - - - - c_0 \gtrdot - \dashv$$

$$\vdash \lessdot - - - - - - - - - - - c \lessdot - - - - - - - - - - - - c_0 \lessdot - - - - - - - c \gtrdot - - - - - - - - - - - - - \dashv$$

$$\vdash \lessdot - - - - - - - - - - - c \lessdot - - - - - - - - - s \gtrdot c_0 \lessdot - - - - - - - c \lessdot - - - - - - - - - - - - s \gtrdot \dashv$$

$$\vdash \lessdot - - - - - - - - - r \gtrdot c \lessdot - - - - - - - r \gtrdot s \gtrdot c_0 \lessdot - - - - - - - c \lessdot c \doteq - - - - - - - r \gtrdot s \gtrdot \dashv$$

$$\vdash \lessdot - - c \doteq - - r \gtrdot r \gtrdot c \lessdot - - - - s \gtrdot r \gtrdot s \gtrdot c_0 \lessdot c \doteq - - r \gtrdot c \lessdot c \lessdot - - c \doteq r \gtrdot r \gtrdot s \gtrdot \dashv$$

$$\vdash \lessdot s \gtrdot c \lessdot s \gtrdot r \gtrdot r \gtrdot c \lessdot c \doteq r \gtrdot s \gtrdot r \gtrdot s \gtrdot c_0 \lessdot c \lessdot s \gtrdot r \gtrdot c \lessdot c \lessdot s \gtrdot c \doteq r \gtrdot r \gtrdot s \gtrdot \dashv$$

$$\vdash \lessdot \underbrace{s}_{u_1} \gtrdot \underbrace{c \lessdot s \gtrdot r}_{w_1} \gtrdot \underbrace{r}_{u_2} \gtrdot \underbrace{c \lessdot c \doteq r \gtrdot s \gtrdot r}_{w_2} \gtrdot \underbrace{s}_{u_3} \gtrdot c_0 \lessdot \underbrace{c \lessdot s \gtrdot r}_{v_1} \gtrdot c \lessdot \underbrace{c \lessdot s \gtrdot c \doteq r \gtrdot r \gtrdot s}_{v_2} \gtrdot \dashv$$

Figure 3: Precedence relations between characters during the parsing of the string of Fig. 1. The dummy string delimiters $\vdash, \dashv$ by hypothesis respectively yield and take precedence over any other character.

(a) $(p_i, \perp\gamma W) \overset{*}{\underset{x}{\mapsto}} (p_h, \perp\gamma W) \iff \langle p_i, p_h \rangle \overset{*}{\Rightarrow} x, x \in \Sigma^*$
(b) $(p_h, \perp\gamma W) \underset{c}{\mapsto} (p_t, \perp\gamma W Z)$
(c) $(p_t, \perp\gamma W Z) \overset{*}{\underset{w_1}{\mapsto}} (p_r, \perp\gamma W Z) \iff \langle p_t, Z, p_r \rangle \overset{*}{\Rightarrow} w_1$
(d) $(p_r, \perp\gamma W Z) \underset{r}{\mapsto} (p_j, \perp\gamma W)$

we derive:

$$(p_i, \perp\gamma W) \overset{*}{\underset{w}{\mapsto}} (p_j, \perp\gamma W) \iff \langle p_i, W, p_j \rangle \overset{*}{\Rightarrow} w, w = xcw_1r \tag{1}$$

Special cases, such as $x = \varepsilon$ and many others, can be similarly treated. □N.B. Each inductive proof of the various assertions may exploit other assertions in the inductive steps. For instance the inductive hypothesis (a) above is based on assertion 3.

A natural question is whether every FG defines a VPDA language or not.

Theorem 3.4. *The VPDA language family is strictly included in the FG family.*

Proof *The language*

$$L = \{b^n c^n \mid n \geq 1\} \cup \{f^n d^n \mid n \geq 1\} \cup \{e^n (fb)^n \mid n \geq 1\}$$

is a FG language but not a VPDA language. L *is generated by the FG grammar*
$S \to A \mid B \mid C \qquad A \to bAc \mid bc \qquad B \to fBd \mid fd \qquad C \to eCfb \mid efb$
which has precedence relations M*:*

$$b \doteq c, f \doteq d, e \doteq f, f \doteq b, b \lessdot b, f \lessdot f, e \lessdot e, c \gtrdot c, d \gtrdot d, b \gtrdot f$$

From $b^n c^n \subseteq L$ *it follows* b *must be a call and* c *a return. For similar reasons,* f *must be a call and* d *a return. From* $e^n (fb)^n \subseteq L$ *it follows that at least one of* b *and* f *must be a return, a contradiction for a VP alphabet.*

FG with a partitioned precedence matrix

We prove that the OPM structure obtained in the proof of Theor. 3.3 is a sufficient condition for an FG to generate a VPDA language thus obtaining a complete characterization of VPDA as a subclass of FG. For an alphabet Σ, let M_T be an OPM such that there exists a partition of Σ into three subsets Σ_1, Σ_2 and, Σ_3 satisfying the conditions:
$\forall a \in \Sigma_1, \forall b \in \Sigma_1 \cup \Sigma_3 : M_T[a,b] = \lessdot$ and $\forall a \in \Sigma_1, \forall b \in \Sigma_2 : M_T[a,b] = \doteq$.
$\forall a \in \Sigma_2, \forall b \in \Sigma : M_T[a,b] = \gtrdot$
$\forall a \in \Sigma_3, \forall b \in \Sigma : M_T[a,b] = \gtrdot$
Then M_T is termed a *total VP-matrix* representing the VP alphabet $\widehat{\Sigma} = \langle \Sigma_1, \Sigma_2, \Sigma_3 \rangle = \langle \Sigma_c, \Sigma_r, \Sigma_i \rangle$, shown in Fig. 2. Any OPM $M \subseteq M_T$ is termed a *VP-matrix*.
Observe that, for any grammar G, such that $OPM(G)$ is a VP-matrix, any production $A \to \alpha$ has $|\alpha|_\Sigma \leq 2$. The possible stencils (or skeletons) of the right parts of the productions are $NcN, NcNr, Nr, Ns$, and those obtained by erasing one or more N. Notice that the stencils rN, crN are forbidden because r does not yield precedence to any character. It follows that, for any FG having a VP matrix, the length of any right part is ≤ 4.

Theorem 3.5. *Let G be an FG such that $OPM(G)$ is a VP matrix. Then $L(G)$ is a VPDA language.*

Proof *First we argue that the grammar generates any string in $L(G)$ with a syntax structure corresponding to the factorization presented in Lemma 3.2. Then, in Lemma 3.6, we construct a VPDA equivalent to G. Let G satisfy the hypotheses of Theor. 3.5. For every string $x \in L(G)$, the syntax tree induces the factorization*
$x = yc_0z$ *or* $x = y, y = u_1w_1u_2w_2 \dots u_kw_k, z = v_1c_1v_2c_2 \dots c_{r-1}v_r$
where all terms are as in Lemma 3.2, and its syntax tree has the structure shown in Fig. 1. It suffices to consider that the precedence relations of the VP matrix completely determine the skeleton of the syntax tree (see Fig. 3).

Lemma 3.6. *Let $G = (\Sigma, V_N, P, S)$ satisfy the hypotheses of Theor. 3.5. Then $L(G)$ is recognized by a VPDA automaton $\mathcal{A} = (\Sigma, Q, Q_0, \Gamma, \delta, Q_F)$, which can be effectively constructed.*

Proof *We specify how to construct from the grammar productions a VPDA $\mathcal{A}$, that recognizes by final state and for convenience is nondeterministic. We recall the production stencils are just the ones previously listed. We set $Q = V_N \cup \{q_0, p, q_F\}$, where $q_0, p, q_F \notin V_N$. The pushdown vocabulary is*

$$\Gamma = \big((V_N \cup \{-\}) \times \Sigma_c \times (V_N \cup \{-\}) \big) \cup \{\perp, Z_U\}$$

Intuitively, $\mathcal{A}$ is built in such a way that it enters a state $B \in V_N$ after finishing the scanning of a substring syntactically rooted in B.
In state B, reading a symbol $c \in \Sigma_c$ (the only ones that yield precedence), $\mathcal{A}$ enters state p and pushes on the stack a symbol, for which two cases occur. The symbol is Z_U, if the c is not to be matched by an r; it is $\langle B, c, C \rangle$, if the machine "looks for" a well-balanced string w such that $C \stackrel{*}{\Rightarrow} w$. Simpler special cases also occur, such that $\mathcal{A}$ pushes on the stack a symbol $\langle B, c, - \rangle$ or $\langle -, c, - \rangle$, "looking" directly for r.
In state p, reading a c, $\mathcal{A}$ remains in the state and pushes on the stack either the symbol Z_U if the c is not to be matched, or a symbol $\langle -, c, C \rangle$ if it "looks for" a string w such that $C \stackrel{*}{\Rightarrow} w$.
Finally we describe the moves that read $r \in \Sigma_r$. If the stack is empty, the machine enters a state A associated to a nonterminal. If the top of stack is a symbol $\langle B, c, C \rangle$, the machine pops the stack and enters a state A. Here too some simpler special cases exist.

The final states set is defined as $Q_F = \{A \mid S \overset{*}{\Rightarrow} \beta A, A \neq S\} \cup \{q_F\} \cup \{q_0 \text{ iff } S \to \varepsilon \in P\}$. *Notice that a production* $A \to cB$ *can be used only in a derivation such as* $S \overset{*}{\Rightarrow} \alpha A \Rightarrow \alpha c B \overset{*}{\Rightarrow} x$, *otherwise* c *would take precedence over some other character. Thus,* A *and* B *are both in* Q_F.

Table 5: Transition relation δ of $\mathcal{A}$.

	productions	δ
1	$A \to s$	(q_0, s, A)
	$A \to r$, such that $S \overset{*}{\Rightarrow} A\alpha$	$(q_0, r, \perp, A)$
2	$A \to s$	(p, s, A)
	$A \to Bs$	(B, s, A)
	$A \to Br$	$(B, r, \perp, A)$
3	$A \to cB$	(p, c, p, Z_U)
	$S \to BcC$	(B, c, p, Z_U)
4	$S \to BcCr$	$(B, c, p, \langle B, c, C\rangle)$
		$(C, r, \langle B, c, C\rangle, q_F)$
	$S \to s$	(q_0, s, q_F)
	$S \to c$	(q_0, c, Z_U, q_F)
	$S \to r$	$(q_0, r, \perp, q_F)$
	$A \to BcCr$	$(B, c, p, \langle B, c, C\rangle)$
		$(p, r, \langle B, c, C\rangle, A)$
	$A \to Bcr$	$(B, c, p, \langle B, c, -\rangle)$
		$(p, r, \langle B, c, -\rangle, A)$
5	$A \to cBr$	$(p, c, p, \langle -, c, B\rangle)$
		$(B, r, \langle -, c, B\rangle, A)$
	$A \to cr$	$(p, c, p, \langle -, c, -\rangle)$
		$(B, r, \langle -, c, -\rangle, A)$

The transition relation δ *is then built from* P *according to Table 5. Notice that the derivations* $S \overset{*}{\Rightarrow} A\alpha$ *needed in section 1 of the table can be effectively computed.*
The proof of the equivalence $L(\mathcal{A}) = L(G)$ *somewhat mirrors the equivalence proof of Theor. 3.3. For instance, from section 2 of Table 5 the following lemma immediately descends:*

$$A \overset{*}{\Rightarrow} w, w \in \left(\Sigma_i \cup \Sigma_r\right)^* \iff \exists \sigma \in (\Gamma \setminus \{\perp\})^*, t \in Q \text{ such that } (t, \perp\sigma) \underset{w}{\overset{*}{\mapsto}} (A, \perp\sigma)$$

Similarly, the lemma

$$A \overset{*}{\Rightarrow} w, w \text{ well balanced } \iff \exists \sigma \in (\Gamma \setminus \{\perp\})^*, t \in Q \text{ such that } (t, \perp\sigma) \underset{w}{\overset{*}{\mapsto}} (A, \perp\sigma)$$

can be proved by a natural induction, taking as the basis the cases $A \to cr$ and $A \to s$, and then exploiting for the induction steps sections 2, 4, and 5 of Table 5. Further details of the proof are omitted as fairly obvious.

Second, we remark that various subclasses of VPDA languages recently considered correspond to restrictions on the VP-precedence matrix and/or on the stencils of the grammar productions. A nice illustration is the family BALAN [2]. First, balanced grammars do not allow any c_i or r_i to be unmatched. Thus an FG such that no production has the stencils $Nc_iN, Nc_i, c_iN, c_i, Nr_i$ ensure the balancing property. Second, balanced grammars do not allow a c_i to be matched by distinct returns r_j, r_k (and similarly for r_i). An FG such that $|\Sigma_c| = |\Sigma_r|$ and the OPM submatrix identified by rows Σ_{c_i} and columns Σ_{r_i} contains $\doteq$ only on the diagonal, ensures the bijection of call and return characters.

4. Closure properties

All families considered here (except DPDA) share the property of being boolean algebras, for suitably defined subsets. The core of the property dates back to the original approach by McNaughton and the ”structure preserving” operations as in [9]. Other closure properties possessed by VPDA, though relevant and classical, have been less investigated. It appears that all the previous families more general than VPDA lack (or are unknown to have) some closure properties, as shown in the next table.

family	boolean operations	concatenation, star	reversal
VPDA [1]	yes for a fixed VP alphabet	yes for a fixed VP alphabet	yes
FG	yes for compatible precedence matrices [9]	probably yes	yes (proved here)
HRDPDA	yes for H-synchronized languages [18]	no [4]	no (proved here)

The reversal of a FG language is generated by the specularly reversed productions; they are a FG grammar with a matrix obtained interchanging yield- and take-precedence relations.
We observe that the boolean closure of FG languages has been proved in [9] by extending McNaughton’s method for parentheses languages. It states that the union of two FG having compatible precedence matrices is a FG language with compatible matrices, and similarly for the other operators. We notice that this is not implied by the closure property [18] of the equivalence class of H-synchronized HDPDA languages, although two FG’s having compatible matrices are necessarily H-synchronized[2].
On the other hand the closure of VPDA languages for a given VP alphabet, under the boolean operators and under reversal, are an immediate consequence of the same properties of the family of FG languages having compatible precedence relations,
Since HRDPDA=RDPDA, the non-closure under reversal follows from a classical counterexample, used for proving the same for deterministic languages: the reversal of $\{1a^nb^n \mid n \geq 0\} \cup \{2a^nb^{2n} \mid n \geq 0\}$ is non-deterministic.
The proof of concatenation and star closures for FG’s is more intricate than with other traditional families of CF grammars due to the need to preserve the operator structure and the precedence relations.[3]
In conclusion, the FG family is currently the one, among the existing VPDA generalizations, that preserves the majority, and possibly the totality, of VPDA closure properties.

[2] For brevity we omit the natural construction of the HDPDA equivalent to a FG grammar.
[3] A complete proof is under development.

5. Conclusions

We mention some open questions raised by the present study.
FG appears at present to be the family that preserves the majority, and possibly the totality, of VPDA closure properties, but we wonder whether more general families can be found with the same properties.

In a different direction, it is possible to transfer to VPDA a rather surprising invariance property of FG. We recall the definition of *Non-Counting context-free* grammar [7], which extends the notion of NC regular language [19]. $L = L(G)$ is NC if for the parenthesized language $L(\widetilde{G})$, the following condition holds: $\exists n > 0 : \forall x, v, w, \underline{v}, y \in \Sigma^*$, where w and $vw\underline{v}$ are well-parenthesized, and $\forall m \geq 0$, $xv^n w\underline{v}^n y \in \widetilde{L}$ if, and only if, $xv^{n+m} w\underline{v}^{n+m} y \in \widetilde{L}$. In general, two equivalent CF grammars may differ with respect to the NC property. However if an FG grammar is NC, then all equivalent FG grammars are NC [8]. Consider now, for a VPDA $L \subseteq \Sigma^*$, two equivalent VPDA recognizers. Notice the two VP alphabets may differ with respect to the 3-partition of the letters. The two corresponding FG's (Theor. 3.3) may differ in precedence relations, but they are either both NC or both counting. We wonder whether such invariance property holds for other families of grammars generalizing VPDA.

Last, it would be interesting to assess the suitability of Floyd languages for the applications that have motivated balanced grammars and VPDA. We observe that the greater generative capacity of FG's permits to define more realistic recursively nested structures. For instance, the VPDA approach uses single characters to represent a call c and the corresponding return r, but this is just an abstraction. In real programming languages a call is a string typically containing the name of the invoked procedure and possibly a list of parameters. Also, at it is suggested by the example in the proof of Theor 3.4, a return corresponding to a given call may use the same characters as some other call. This will cause conflicts in the partitioning of Σ, but can be dealt with by suitable precedence relations. Similar examples can be found in the area of mark-up languages.
Finally, for application in model checking, the computational complexity of the decision problems for FG languages should be studied.

References

[1] R. Alur and P. Madhusudan. Visibly pushdown languages. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2004.

[2] J. Berstel and L. Boasson. Balanced grammars and their languages. In W. Brauer et al., editor, *Formal and Natural Computing*, volume 2300 of *LNCS*, pages 3–25. Springer, 2002.

[3] D. Caucal. Synchronization of pushdown automata. In O. H. Ibarra and Z. Dang, editors, *Developments in Language Theory*, volume 4036 of *LNCS*, pages 120–132. Springer, 2006.

[4] D. Caucal. Boolean algebras of unambiguous context-free languages. In R. Hariharan, M. Mukund, and V. Vinay, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008)*, Dagstuhl, Germany, 2008. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[5] D. Caucal and S. Hassen. Synchronization of grammars. In Edward A. Hirsch, Alexander A. Razborov, Alexei L. Semenov, and Anatol Slissenko, editors, *CSR*, volume 5010 of *LNCS*, pages 110–121. Springer, 2008.

[6] S. Crespi-Reghizzi. An effective model for grammar inference. In *Information Processing 71*, pages 524–529. North-Holland Publishing Co., 1972.

[7] S. Crespi-Reghizzi, G. Guida, and D. Mandrioli. Noncounting context-free languages. *JACM: Journ. of the ACM*, 25:571–580, 1978.

[8] S. Crespi-Reghizzi, G. Guida, and D. Mandrioli. Operator precedence grammars and the noncounting property. *SICOMP: SIAM Journ. on Computing*, 10:174—191, 1981.

[9] S. Crespi-Reghizzi, D. Mandrioli, and D. F. Martin. Algebraic properties of operator precedence languages. *Information and Control*, 37(2):115–133, May 1978.

[10] M. J. Fischer. Some properties of precedence languages. In *STOC '69: Proc. first annual ACM symp.m on Theory of computing*, pages 181–190, New York, NY, USA, 1969. ACM.

[11] D. Fisman and A. Pnueli. Beyond regular model checking. *FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, 21, 2001.

[12] R. W. Floyd. Syntactic analysis and operator precedence. *J. ACM*, 10(3):316–333, 1963.

[13] D. Grune and C. J. Jacobs. *Parsing techniques: a practical guide*. Springer, New York, 2008.

[14] M. A. Harrison. *Introduction to Formal Language Theory*. Addison Wesley, 1978.

[15] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata and Formal Languages*. Addison-Wesley, Reading, MA, 1979.

[16] N. Limaye, M. Mahajan, and A. Meyer. On the complexity of membership and counting in height-deterministic pushdown automata. In E.A. Hirsch et al., editor, *CSR*, volume 5010 of *LNCS*, pages 240–251. Springer, 2008.

[17] R. McNaughton. Parenthesis grammars. *J. ACM*, 14(3):490–500, 1967.

[18] D. Nowotka and J. Srba. Height-deterministic pushdown automata. In Ludek Kucera and Antonín Kucera, editors, *MFCS*, volume 4708 of *LNCS*, pages 125–134. Springer, 2007.

[19] R. McNaughton and S. Papert. *Counter-free Automata*. MIT Press, Cambridge, USA, 1971.

[20] J. W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journ. of Comp. and Syst.Sc.*, 1:317–322, 1967.

[21] S. La Torre, M. Napoli, and M. Parente. On the membership problem for visibly pushdown languages. In S. Graf and W. Zhang, editors, *ATVA Automated Technology for Verification and Analysis*, volume 4218 of *LNCS*, pages 96–109. Springer, 2006.