Journal of Universal Computer Science, vol. 1, no. 1 (1995), 67-82
submitted: 8/11/94, accepted: 10/1/95, appeared: 28/1/95©Springer Pub. Co.

Grammars Based on the Shuffle Operation

Gheorghe Păun
Institute of Mathematics of the Romanian Academy of Sciences
PO Box 1 – 764, Bucureşti, Romania

Grzegorz Rozenberg
University of Leiden, Department of Computer Science
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Arto Salomaa
Academy of Finland and University of Turku
Department of Mathematics, 20500 Turku, Finland

Abstract: We consider generative mechanisms producing languages by starting from a finite set of words and shuffling the current words with words in given sets, depending on certain conditions. Namely, regular and finite sets are given for controlling the shuffling: strings are shuffled only to strings in associated sets. Six classes of such grammars are considered, with the shuffling being done on a leftmost position, on a prefix, arbitrarily, globally, in parallel, or using a maximal selector. Most of the corresponding six families of languages, obtained for finite, respectively for regular selection, are found to be incomparable. The relations of these families with Chomsky language families are briefly investigated.

Key Words: Shuffle operation, Chomsky grammars, L Systems

Categories: F4.2 [Mathematical Logic and Formal Languages]: Grammars and other Rewriting Systems: *Grammar types*, F4.3 [Mathematical Logic and Formal Languages]: Formal Languages: *Operations on languages*

1 Introduction

In formal language theory, besides the basic two types of grammars, the Chomsky grammars and the Lindenmayer systems, there are many "exotic" classes of generative devices, based not on the process of rewriting symbols (or strings) by strings, but using various operations of adjoining strings. We quote here the string adjunct grammars in [7], the semi-contextual grammars in [3], the ξ-grammars in [10], the pattern grammars in [2], and the contextual grammars [8]. The starting point of the present paper is this last class of grammars, via the paper [9], where a variant of contextual grammars was introduced based on the shuffle operation. Basically, one gives two finite sets of strings, B and C, over some alphabet, and one considers the set of strings obtained by starting from B and iteratively shuffling strings from C, without any restriction. This corresponds to the simple contextual grammars in [8].

We consider here a sort of couterpart of the contextual grammars with choice, where the adjoining is controlled by a selection mapping. In fact, we proceed in a way similar to that used in conditional contextual grammars [12], [13], [14] and in modular contextual grammars [15]: we start with several pairs of the form (R_i, C_i), R_i a language (we consider here only the case when R_i is regular or finite) and C_i a finite set of strings, and allow the strings in C_i to be shuffled only to strings in R_i. Depending on the place of the string in R_i in the current string generated by our grammar, we can distinguish several types of grammars: prefix (the string in R_i appears in the left-hand of the processed string, as a prefix of it), leftmost (we look for the leftmost possible occurrence of a string in R_i), arbitrary (no condition on the place where the string in R_i appears), global (the whole current string is in R_i), and parallel (the current string is portioned into strings in R_i). An interesting variant is to use a substring as base for shuffling only when it is maximal. Twelve families of languages are obtained in this way. Their study is the subject of this paper.

It is worth noting that the shuffle operation appears in various contexts in algebra and in formal language theory; we quote only [1], [4], [5], [6] (and [11], for applications). In [1], [6] the operation is used in a generative-like way, for identifying families of languages of the form $FAM(\cup, \cdot, Shuf; FIN)$, the smallest family of languages containing the finite languages and closed under union, concatenation and shuffle. (From this point of view, the family investigated in [9] is a particular case, $FAM(Shuf; FIN)$).

The shuffle of the symbols of two words is also related to the concurrent execution of two processes described by these words, hence our models can be interpreted in terms of concurrent processes, too. For instance, the prefix mode of work corresponds to the concurrent execution of a process strictly at the beginning of another process, the "beginning" being defined modulo a regular language; in the global case the elementary actions of the two processes can be freely intercalated.

As it is expected, the languages generated by shuffle grammars are mostly incomparable with Chomsky languages (due to the fact that we do not use nonterminals). Somewhat surprising is the fact that in the regular selection case the five modes of work described above, with only one exception, cannot simulate each other (the corresponding families of languages are incomparable).

2 Classes of shuffle grammars

As usual, V^* denotes the set of all words over the alphabet V, the empty word is denoted by λ, the length of $x \in V^*$ by $|x|$ and the set of non-empty words over V is identified by V^+. The number of occurrences of a symbol a in a string x will be denoted by $|x|_a$. For basic notions in formal language theory (Chomsky grammars and L systems) we refer to [16], [17]. We only mention that $FIN, REG, CF, CS, 0L$ are the families of finite, regular, context-free, context-sensitive, and of 0L languages, respectively.

For $x, y \in V^*$ we define the *shuffle* (product) of x, y, denoted $x \sqcup\!\sqcup y$, as

$$x \sqcup\!\sqcup y = \{x_1y_1x_2y_2 \dots x_ny_n \mid x = x_1x_2 \dots x_n, y = y_1y_2 \dots y_n, \\ x_i, y_i \in V^*, 1 \leq i \leq n, n \geq 1\}.$$

Various properties of this operation, such as commutativity and associativity, will be implicitely used in the sequel. The operation $\sqcup\!\sqcup$ is extended in the natural way to languages,

$$L_1 \sqcup\!\sqcup L_2 = \{z \mid z \in x \sqcup\!\sqcup y, x \in L_1, y \in L_2\},$$

and iterated,

$$L^{(0)} = \{\lambda\},$$
$$L^{(i+1)} = L^{(i)} \sqcup\!\sqcup L, \; i \geq 0,$$
$$L^{\sqcup\!\sqcup} = \bigcup_{i \geq 0} L^{(i)}.$$

The grammars considered in [9] are triples of the form $G = (V, B, C)$, where V is an alphabet, B and C are finite languages over V. The language generated by G is defined as the smallest language L over V containing B and having the property that if $x \in L$ and $u \in C$, then $x \sqcup\!\sqcup u \subseteq L$. (Therefore, this language is equal to $B \sqcup\!\sqcup C^{\sqcup\!\sqcup}$).

There is no restriction in [9] about the shuffling of elements of C to current strings. Such a control of the grammar work can be done in various ways of introducing a context-dependency. We use here the following natural idea:

Definition 1. A *shuffle grammar* is a construct

$$G = (V, B, (R_1, C_1), \dots, (R_n, C_n)),$$

where V is an alphabet, B is a finite language over V, R_i are languages over V and C_i are finite languages over V, $1 \leq i \leq n$.

The parameter $n \geq 1$ is called the *degree* of G. If R_i are languages in a given family F, then we say that G is *with F choice*. Here we consider only the cases $F = FIN$ and $F = REG$.

The idea is to allow the strings in C_i to be shuffled only to strings in the corresponding set R_i. The sets R_i are called *selectors*.

Definition 2. For a shuffle grammar G as above, a constant $i, 1 \le i \le n$, and two strings x, y in V^*, we define the following derivation relations:

$$\begin{aligned}
x \Longrightarrow_i^{arb} y \; iff \; & x = x_1x_2x_3, x_1, x_3 \in V^*, x_2 \in R_i, y = x_1x_2'x_3, \\
& for\ some\ x_2' \in x_2 ⧢ u, u \in C_i, \\
x \Longrightarrow_i^{pr} y \; iff \; & x = x_1x_2, x_2 \in V^*, x_1 \in R_i, y = x_1'x_2, \\
& for\ some\ x_1' \in x_1 ⧢ u, u \in C_i, \\
x \Longrightarrow_i^{lm} y \; iff \; & x = x_1x_2x_3, x_1, x_3 \in V^*, x_2 \in R_i, y = x_1x_2'x_3, \\
& for\ some\ x_2' \in x_2 ⧢ u, u \in C_i,\ and\ there\ is\ no\ j, 1 \le j \le n, \\
& such\ that\ x = v_1v_2v_3, |v_1| < |x_1|, v_2 \in R_j, \\
x \Longrightarrow_i^{gl} y \; iff \; & x \in R_i, y = x ⧢ u,\ for\ some\ u \in C_i.
\end{aligned}$$

These derivation relations are called *arbitrary, prefix, leftmost, and global* derivations, respectively. Moreover, we define the *parallel* derivation as

$$\begin{aligned}
x \Longrightarrow^{pl} y \; iff \; & x = x_1x_2 \ldots x_k, k \ge 1, y = x_1'x_2' \ldots x_k', \\
& for\ x_i \in R_{j_i}, x_i' \in x_i ⧢ u_i, u_i \in C_{j_i}, 1 \le j_i \le n, 1 \le i \le k.
\end{aligned}$$

We denote $M = \{arb, pr, lm, gl, pl\}$.

Definition 3. The *language generated* by a shuffle grammar G in the mode $f \in M - \{pl\}$ is defined as follows:

$$\begin{aligned}
L_f(G) = B \cup \{x \in V^* \mid w \Longrightarrow_{i_1}^f w_1 \Longrightarrow_{i_2}^f \ldots \Longrightarrow_{i_m}^f w_m = x, \\
w \in B, 1 \le i_j \le n, 1 \le j \le m, m \ge 1\}.
\end{aligned}$$

For the parallel mode of derivation we define

$$L_{pl}(G) = B \cup \{x \in V^* \mid w \Longrightarrow^{pl} w_1 \Longrightarrow^{pl} \ldots \Longrightarrow^{pl} w_m = x, w \in B, m \ge 1\}.$$

The corresponding families of languages generated by shuffle grammars with F choice, $F \in \{FIN, REG\}$, are denoted by $ARB(F), PR(F), LM(F), GL(F), PL(F)$, respectively; by SL we denote the family of languages generated by grammars as in [9], without choice. Subscripts n can be added to ARB, PR, LM, GL, PL when only languages generated by grammars of degree at most $n, n \ge 1$, are considered.

3 The generative capacity of shuffle grammars

From definition we have the inclusions $X_n(F) \subseteq X_{n+1}(F)$, $X_n(FIN) \subseteq X_n(REG)$, for all $n \ge 1$, $X \in \{ARB, PR, LM, GL, PL\}$ and $F \in \{FIN, REG\}$.

Every family $ARB(FIN), PR(FIN), LM(FIN), GL(FIN), PL(FIN)$ contains each finite language. This is obvious, because for $G = (A, B, (B, \{\lambda\}))$ we have $L_f(G) = B$ for all f. In fact, we have

Theorem 1.
(i) *Every family $ARB_1(REG), PR_1(REG), LM_1(REG), GL_1(REG)$, $PL_1(REG)$ includes strictly the family SL.*
(ii) *The family SL is incomparable with each $X_n(FIN), n \ge 1, X \in \{ARB, PR, LM, PL\}$.*
(iii) *$GL(FIN) = GL_1(FIN) = FIN$.*

Proof.
(i) If $G = (V, B, C)$ is a simple shuffle grammar, then $L(G) = L_f(G')$ for each $f \in M$ and

$$G' = (V, B, (V^*, C)).$$

(The only point which needs some discussion is the fact that a parallel derivation $x \Longrightarrow^{pl} y$ in G', with $x = x_1x_2 \ldots x_k$ and $y = x'_1x'_2 \ldots x'_k, k\ ge2$, can be simulated by a k-step derivation in G, because the strings shuffled into $x_1, \ldots, x_k$ do not overlap each other.) Consequently, $SL \subseteq X_1(REG)$, for all X.

The inclusion is proper in view of the following necessary condition for a language to be in SL (Lemma 10 in [9]): if $L \in SL$ and

$$V_0 = \{a \in V \mid \ for\ every\ n \geq 1\ there\ is\ x \in L\ with\ |x|_a \geq n\},$$

then each string in V_0^* is a subword of a string in L. This condition rejects languages such as $L = a^+ \cup b^+$; this language can be generated by the shuffle grammar with finite choice

$$G = (\{a, b\}, \{a, b\}, (\{a\}, \{\lambda, a\}), (\{b\}, \{\lambda, b\}))$$

in all modes of derivation excepting the global one. For gl we take

$$G' = (\{a, b\}, \{a, b\}, (V^*, \{a\}), (b^*, \{b\})).$$

(ii) We have already seen that $X_1(FIN) - SL \neq \emptyset$ for $X \in \{ARB, PR, LM, PL\}$. Consider now the simple shuffle grammar

$$G = (\{a, b, c\}, \{abc\}, \{abc\}).$$

We have

$$L(G) \cap a^+b^+c^+ = \{a^mb^mc^m \mid m \geq 1\}.$$

Assume that $L(G) = L_f(G')$ for some shuffle grammar $G' = (\{a, b, c\}, B, (R_1, C_1), \ldots \ldots, (R_n, C_n))$ with finite $R_1, \ldots, R_n$ and $f \in \{arb, pr, lm\}$. Take a string $z = a^mb^mc^m$ in $L(G)$ with arbitrarily large m. A derivation $w \Longrightarrow_i^f z$ must be possible in G', with $w = a^pb^pc^p$, $p < m$, for some $1 \leq i \leq n$. We must have $w = w_1w_2w_3, w_2 \in R_i$, and $z = w_1w'_2w_3$ for $w'_2 \in w_2 ⧢ u, u \in C_i$. If $u = \lambda$, this derivation step can be omitted, hence we may assume that $u \neq \lambda$.

The sets R_i, C_i are finite. Denote

$$r = \max\{|x| \mid x \in R_i, 1 \leq i \leq n\},$$
$$q = \max\{|x| \mid x \in C_i, 1 \leq i \leq n\}.$$

Therefore $|w_2| \leq r, |u| \leq q$, that is $p \geq m - q$. For $m > r + q$ we have $p \geq m - q > r$, hence either $w_2 \in sub(a^pb^{p-1})$ or $w_2 \in sub(b^{p-1}c^p)$. In both cases, the shuffling with u modifies at most two of the three subwords a^p, b^p, c^p, hence a parasitic string is obtained. (In the prefix derivation case we precisely know that only a^p is modified.)

Consider now the parallel case. Take $G' = (\{a, b, c\}, B, (R_1, C_1), \ldots, (R_n, C_n))$ such that $L_{pl}(G') = L(G), R_i$ finite sets. For obtaining a string $a^mb^mc^m$ with large enough m we need a derivation $a^pb^pc^p \Longrightarrow^{pl} a^mb^mc^m, p < m$. This implies we have sets R_i containing strings $a^r, r \geq 1$ and with the corresponding sets C_i containing strings $a^q, q \geq 1$ (similarly for b and c).

Assume that we find in the sets R_i two strings a^{r_1}, a^{r_2} and in the associated sets C_i we find two strings a^{q_1}, a^{q_2}. Similarly, we have pairs (b^{r_3}, b^{q_3}) and (c^{r_4}, c^{q_4}). The string $a^tb^tc^t$ for $t = r_1r_2r_3r_4$ is in $L(G)$ and it can be rewritten using the same pairs of strings containing the symbols b and c and different pairs for a:

$$a^tb^tc^t \Longrightarrow^{pl} a^{(t/r_1)(r_1+q_1)}b^sc^s,$$
$$a^tb^tc^t \Longrightarrow^{pl} a^{(t/r_2)(r_2+q_2)}b^sc^s,\ for\ some\ s.$$

Consequently, we must have

$$\frac{t}{r_1}(r_1 + q_1) = \frac{t}{r_2}(r_2 + q_2),$$

which implies

$$\frac{q_1}{r_1} = \frac{q_2}{r_2}.$$

For all such pairs (a^r, a^q) we obtain the same value for $\frac{q}{r}$. Denote it by α. Rewriting some $a^s b^s c^s$ using such pairs we obtain

$$\frac{s}{r}(r+q) = s(1+\alpha)$$

occurrences of a. Continuing $k \geq 1$ steps, we get $s(1+\alpha)^k$ occurrences of a, hence a geometrical progression, starting at $a^s b^s c^s$.

Symbols a can be also introduced in a derivation $a^p b^p c^p \Longrightarrow^{pl} a^m b^m c^m$ by using pairs $(a^i b^j, z)$ with x containing occurrences of a. At most one such pair can be used in a derivation step. Let h be the largest number of symbols a in strings z as above (the number of such strings is finite). Thus, in a derivation $a^p b^p c^p \Longrightarrow^{pl} a^m b^m c^m$, the number m can be modified by such pairs in an interval $[m_1, m_2]$ with $m_2 - m_1 \leq h$.

We start from at most $g = card(A)$ strings of the form $a^s b^s c^s$, hence we have at most g geometrical progressions $s(1+\alpha)^k, k \geq 1$. The difference $\beta_k = s(1+\alpha)^{k+1} - s(1+\alpha)^k$ can be arbitrarily large. When $\beta_k > gh$, the at most g progressions can have an element between $s(1+\alpha)^{k+1}$ and $s(1+\alpha)^k$; each such element can have at most h values. Consequently, at least one natural number t between $s(1+\alpha)^{k+1}$ and $s(1+\alpha)^k$ is not reached, the corresponding string $a^t b^t c^t$, although in $L(G)$, is not in $L_{pl}(G')$. This contradiction concludes the proof of point (ii).

(iii) As only strings in the sets $R_i, 1 \leq i \leq n$, of a grammar $G = (V, B, (R_1, C_1), \ldots, (R_n, C_n))$ can be derived, we have $GL(FIN) \subseteq FIN$. The inclusion $FIN \subseteq GL_1(FIN)$ has been pointed out at the beginning of this section. ◊

Corollary. *The families $X_n(REG), n \geq 1, X \in \{ARB, LM, PR, PL, GL\}$, contain non-context-free languages.*

For $PL(FIN), PL(REG)$ and $GL(REG)$ this assertion can be strenghtened.

Theorem 2. *Every propagating unary 0L language belongs to the family $PL_1(FIN)$.*

Proof. For a unary 0L system $G = (\{a\}, w, P)$ with $w \in V^*$ and

$$P = \{a \rightarrow a^{i_1}, a \rightarrow a^{i_2}, \ldots, a \rightarrow a^{i_r}\},$$

with $i_j \geq 1, 1 \leq j \leq r$, we construct the shuffle grammar

$$G' = (\{a\}, \{w\}, (\{a\}, \{a^{i_1 - 1}, a^{i_2 - 1}, \ldots, a^{i_r - 1}\})).$$

It is easy to see that $L(G) = L_{pl}(G')$. ◊

This implies that $PL(FIN), PL(REG)$ contain one-letter non-regular languages. This is not true for the other modes of derivation.

Theorem 3. *A one-letter language is in $ARB(F), PR(F), LM(F), F \in \{FIN, REG\}$, or in $GL(REG)$ if and only if it is regular.*

Proof. Take a shuffle grammar $G = (\{a\}, B, (R_1, C_1), \ldots, (R_n, C_n))$ with regular sets $R_i, 1 \leq i \leq n$. Clearly, $L_{arb}(G) = L_{pr}(G) = L_{lm}(G)$. Because we work with one-letter strings, a string x can be derived using a component (R_i, C_i) if (and only if) the shortest string in R_i is contained in x, hence is of length at most $|x|$. Therefore we can replace each R_i by a^{k_i}, where

$$k_i = \min\{|x| \mid x \in R_i\}, 1 \leq i \leq n,$$

without modifying the generated language. Denote

$$K = \max\{k_i \mid 1 \leq i \leq n\}$$

and construct the right-linear grammar $G' = (V_N, \{a\}, S, P)$ with

$$\begin{aligned}
V_N &= \{S, X\} \cup \{[p] \mid 0 \leq p \leq K\},\\
P &= \{S \to a^p[p] \mid a^p \in B, p < K\} \cup\\
&\cup \{S \to a^p X \mid a^p \in B, p \geq K\} \cup\\
&\cup \{S \to a^p \mid a^p \in B\} \cup\\
&\cup \{[p] \to a^s[p+s] \mid p \leq K, p+s \leq K, p \geq k_i, a^s \in C_i, 1 \leq i \leq n\} \cup\\
&\cup \{[p] \to a^s \mid p \leq K, p \geq k_i, a^s \in C_i, 1 \leq i \leq n\} \cup\\
&\cup \{[p] \to a^s X \mid p \leq K, p+s \geq K, p \geq k_i, a^s \in C_i, 1 \leq i \leq n\} \cup\\
&\cup \{X \to a^s X \mid a^s \in C_i, \ for\ some\ 1 \leq i \leq n\} \cup\\
&\cup \{X \to a^s \mid a^s \in C_i, \ for\ some\ 1 \leq i \leq n\}.
\end{aligned}$$

At the first steps of a derivation in G', the nonterminals $[p]$ count the number of symbols a introduced at that stage, in order to ensure the correct simulation of components of G; when this number exceeds K, then each component can be used, and this is encoded by the nonterminal X. Thus we have the equalities $L(G') = L_f(G), f \in \{arb, pr, lm\}$.

For the global derivation we start from an arbitrary shuffle grammar $G = (\{a\}, B, (R_1, C_1), \ldots, (R_n, C_n))$, with regular sets $R_i, 1 \leq i \leq n$, take for each R_i a deterministic finite automaton $A_i = (Q_i, \{a\}, s_{0,i}, F_i, \delta_i)$, $1 \leq i \leq n$, and construct the right-linear grammar $G' = (Q_1 \times Q_2 \times \ldots \times Q_n, \{a\}, (s_{0,1}, \ldots, s_{0,n}), P)$ with P containing the following rules:

$$\begin{aligned}
&(s_{0,1}, \ldots, s_{0,n}) \to a^r, a^r \in B,\\
&(s_{0,1}, \ldots, s_{0,n}) \to a^r(s_1, \ldots, s_n), \ for\ a^r \in B, s_i = \delta_i(s_{0,i}, a^r), 1 \leq i \leq n,\\
&(s_1, \ldots, s_n) \to a^r(s'_1, \ldots, s'_n), \ for\ s'_j = \delta_j(s_j, a^r), s_j \in Q_j, 1 \leq j \leq n,\\
&\qquad and\ a^r \in C_i, s_i \in F_i, \ for\ some\ i \in \{1, \ldots, n\},\\
&(s_1, \ldots, s_n) \to a^r, \ for\ s_j \in Q_j, 1 \leq j \leq n, \ and\ a^r \in C_i, s_i \in F_i,\\
&\qquad for\ some\ i \in \{1, \ldots, n\}.
\end{aligned}$$

New occurrences of a, corresponding to sets C_i, are added (it does not matter where) only when the current string belongs to R_i, and this is checked by the simultaneous parsings of the current string in the deterministic automata recognizing $R_1, \ldots, R_n$. Consequently, $L_{gl}(G)$ is a regular language.

Conversely, each one-letter regular language L is known to be equal with the disjoint union of a finite language F and a finite set of languages of the form

$$L_i = \{a^m \mid m = p_i + jq, j \geq 0\}.$$

(The constant p_i is associated to L_i, but q is the same for all L_i.) Assume we have n languages L_i; denote

$$K = \max\{t \mid t = p_i, 1 \leq i \leq n, \ or\ a^t \in F\}.$$

Then $L = L_f(G), f \in \{arb, pr, lm\}$, for the grammar

$$G = (\{a\}, \{a^s \in L \mid s \leq K + q\}, (\{a^{K+1}\}, \{a^q\})),$$

hence we have the theorem for $ARB(F), PR(F), LM(F), F \in \{FIN, REG\}$.

For the global case, starting from $L \subseteq a^*$ regular, written as above $L = F \cup \bigcup_{i=1}^n L_i$, $L_i = \{a^m \mid m = p_i + j \cdot q, j \geq 0\}$, $1 \leq i \leq n$, we consider the grammar

$$G = (\{a\}, \{a^s \in L \mid s \leq K + q\}, (\{a^m \mid a^{m+q} \in L\}, \{a^q\})).$$

The equality $L = L_{gl}(G)$ is obvious and this concludes the proof. $\Diamond$

In view of the previous result, it is somewhat surprising to obtain

Theorem 4. *The family GL(REG) contains non-semilinear languages (even on the alphabet with two symbols only).*

Proof. Consider the shuffle grammar $G = (\{a, b, c\}, \{abc\}, (R_1, C_1), \ldots, (R_8, C_8))$, with the following eight components:

$$\begin{array}{ll}
R_1 = (a^2b^2c^2)^*abc(abc)^*, & C_1 = \{a\},\\
R_2 = (a^2b^2c^2)^*a^2bc(abc)^*, & C_2 = \{b\},\\
R_3 = (a^2b^2c^2)^*a^2b^2c(abc)^*, & C_3 = \{c\},\\
R_4 = (abc)^*a^2b^2c^2(a^2b^2c^2)^*, & C_4 = \{b\},\\
R_5 = (abc)^*abab^2c^2(a^2b^2c^2)^*, & C_5 = \{c\},\\
R_6 = (abc)^*abcab^2c^2(a^2b^2c^2)^*, & C_6 = \{c\},\\
R_7 = (abc)^*abcabcbc^2(a^2b^2c^2)^*, & C_7 = \{a\},\\
R_8 = (abc)^*abcabcabcc(a^2b^2c^2)^*, & C_8 = \{ab\}.
\end{array}$$

Examine the derivations in G in the global mode. The whole current string must be in some R_i in order to continue the derivation. Assume we start from a string $w = (abc)^r$ (initially we have $r = 1$). This string is in R_1 only, hence we can add one more occurrence of a. This can be done in all possible positions of w and we obtain a string in $L_{gl}(G)$, but only when we obtain $aabc(abc)^{r-1}$ we can continue the derivation, namely by using the second component of G. Now a symbol b can be added, and again the only case which does not block the derivation leads to $aabbc(abc)^{r-1}$. We can continue with the third component and either we block the derivation, or we get $a^2b^2c^2(abc)^{r-1}$. From a string of the form $(a^2b^2c^2)^+(abc)^*$ we can continue only with the first component, hence the previous operations are iterated. When all symbols a, b, c are doubled, we obtain the string $w' = (a^2b^2c^2)^r$.

Note that $|w'| = 2|w|$ (more precisely, $|w'|_a = 2|w|_a, |w'|_b = 2|w|_b, |w'|_c = |w|_c$), but for each intermediate string z in this derivation at least one of the following inequalities holds: $|z|_a \neq |z|_b, |z|_b \neq |z|_c, |z|_a \neq |z|_c$.

To a string of the form of w' above only the fifth component of G can be applied and again either the derivation must be finished, or it continues only in (R_6, C_6). The derivation proceeds deterministically through $(R_6, C_6), (R_7, C_7), (R_8, C_8)$, inserting step by step new symbols a, b, c between pairs of such symbols in order to obtain again triples abc. In this way we obtain either strings from which we cannot continue or we reach a string of the same form with w, namely $(abc)^{2r}$. The number of a, b, c occurrences has been doubled again, whereas in the intermediate steps we have strings with different numbers of occurrences of at least two of a, b, c.

The process can be iterated. From the previous discussion one can see that for all strings $w \in L_{gl}(G)$ with $|w|_a = |w|_c$ we also have $|w|_a = |w|_b$ and $|w|_a = 2^n, n \geq 0$. Consequently, denoting by $\Psi_{\{a,b,c\}}$ the Parikh mapping with respect to the alphabet $\{a, b, c\}$, we have

$$\Psi_{\{a,b,c\}}(L_{gl}(G)) \cap \{(n, m, n) \mid n, m \geq 1\} = \{(2^n, 2^n, 2^n) \mid n \geq 1\}.$$

This set is not semilinear; the set of semilinear vectors of given dimension is closed under intersection, therefore $L_{gl}(G)$ is not a semilinear language.

Consider now the morphism $h : \{a, b, c\}^* \longrightarrow \{a, b\}^*$ defined by $h(a) = bab, h(b) = baab, h(c) = baaab$. Define the grammar

$$G' = (\{a, b\}, \{h(abc)\}, (h(R_1), h(C_1)), \ldots, (h(R_8), h(C_8))).$$

Shuffling a string $h(a), h(b), h(c)$ with a string $h(z), z \in R_i$, in a way different from inserting $h(a), h(b), h(c)$ as a block leads to strings with substrings of the forms bbb or aba. Take, for example, $h(z) = z_1bbabbz_2$ and examine the possibilities to shuffle $h(b) = baab$ after z_1. In order to not get a substring bbb we must insert the first symbol of $h(b)$ after the specified occurrence of a in $h(z)$. If we continue with the symbol b of $h(z)$, this is as starting after this occurrence of b, if we continue with the symbol a of $h(b)$, then we obtain aba. Starting after bab in $h(z)$, after introducing one b from $h(b)$ we either get bbb (if we continue in $h(z)$), or $baba$ (if we alternate in $h(b), h(z), h(b)$), or $baab$ (hence $h(b)$ is inserted as a block). Similarly, take for example $h(z) = z_1bbaaabbz_2$ and shuffle the same $h(b) = baab$. Introducing b after z_1bba we have to continue either with a from $h(z)$ or with a from $h(b)$, in both cases obtaining the subword aba. The same result is obtained in all other cases. After producing a substring bbb or aba, the derivation is blocked. Inserting $h(a), h(b), h(c)$ in $h(z)$ as a compact block corresponds to inserting a, b, c in z. Consequently, the derivations in G' correspond to derivations in G. When a derivation is blocked, it can terminate with a string z with $|z|_a = |z|_b$ only when z contains the same number of substrings bab abd $baaab$, with the possible exception of such substrings destroyed by the last shuffling, that of $baaab$ when using the pair $(h(R_3), h(C_3))$ or of $babbaab$ when using the pair $(h(R_8), h(C_8))$.

Moreover,

$$\Psi_{\{a,b\}}(L_{gl}(G')) \cap \{(n, n) \mid n \geq 1\} = \{(6 \cdot 2^n, 6 \cdot 2^n) \mid n \geq 1\}.$$

Indeed, from a string z containing 2^n occurrences of every symbol a, b, c we get a string $h(z)$ with

$$2^n + 2 \cdot 2^n + 3 \cdot 2^n = 6 \cdot 2^n \textit{ occurrences of } a,$$
$$2 \cdot 2^n + 2 \cdot 2^n + 2 \cdot 2^n = 6 \cdot 2^n \textit{ occurrences of } b.$$

Conversely, if a string $h(z)$ contains the same number of a and b occurrences, assume that it contains α substrings bab, β substrings $baab$ and γ substrings $baaab$. Then it contains $2\alpha + 2\beta + 2\gamma$ occurrences of b and $\alpha + 2\beta + 2\gamma$ occurrences of b. Consequently,

$$2\alpha + 2\beta + 2\gamma = \alpha + 2\beta + 2\gamma$$

which implies $\alpha = \gamma$. As we have seen in the first part of the proof, when $|w|_a = |w|_c$, then also $|w|_a = |w|_b$, hence $\alpha = \beta$, too. Therefore the obtained string x corresponds to a string z (in the sense $x = h(z)$) such that $|z|_a = |z|_b = |z|_c$. This implies $|x|_a = |x|_b$ and $|x|_a = |x|_b = 6 \cdot 2^n$.

In conclusion, also $L_{gl}(G')$ is not semi-linear. ◊

For the case of finite selection we have

Theorem 5. $PR(FIN) \subseteq REG$.

Proof. Let $G = (V, B, (R_1, C_1), \ldots, (R_n, C_n))$ be a shuffle grammar with all sets R_i, $1 \leq i \leq n$, finite. We construct a left-linear grammar $G' = (N, V, S, P)$ as follows.

Let

$$q = \max\{|x| \mid x \in R_i, 1 \leq i \leq n, \textit{ or } x \in B\}.$$

Then

$$\begin{aligned}
N = \; & \{[x] \mid x \in V^*, |x| \leq q\} \cup \{S\},\\
P = \; & \{S \to [x] \mid x \in B\} \cup\\
& \cup\{[w] \to [z] \mid \textit{ if } |w| \leq q, w = w_1w_2, w_1 \in R_i \textit{ for some } 1 \leq i \leq n,\\
& \quad z = uw_2, \textit{ for some } u \in w_1 ⧢ x, x \in C_i \textit{ and } |z| \leq q\} \cup\\
& \cup\{[w] \to [z]v \mid \textit{ if } |w| \leq q, w = w_1w_2, w_1 \in R_i \textit{ for some } 1 \leq i \leq n,\\
& \quad zv = uw_2 \textit{ for some } u \in w_1 ⧢ x, x \in C_i \textit{ and } |z| = q\} \cup\\
& \cup\{[w] \to w \mid |w| \leq q\}.
\end{aligned}$$

The derivation proceeds from right to left; the nonterminals $[w]$ in the left-hand side of sentential forms memorize the prefix w of large enough length to control the derivation in G in the prefix mode. Therefore, $L_{pr}(G) = L(G')$ and $L_{pr}(G) \in REG$. ◊

Theorem 6. $ARB_1(FIN)$ *contains non-linear languages.*

Proof. Take $G = (\{a, b\}, \{\lambda\}, (\{\lambda\}, \{ab\}))$. We obviously have $L_{arb}(G) =$ the Dyck language over $\{a, b\}$, known to be a (context-free) non-linear language. ◊

Open problems. Are there non-semilinear languages in families $ARB(F), LM(F), F \in \{FIN, REG\}$, or in $PR(REG)$? Are there non-context-free languages in families $ARB(FIN), LM(FIN)$?

We proceed now to investigating the relations among families $ARB(F), PR(F), LM(F), GL(F), PL(F)$ for F as above. We present the results in the next theorem, whose proof will consist of the series of lemmas following it.

Theorem 7. (i) *Each two of the families* $ARB(REG), PR(REG)$, $LM(REG)$, $GL(REG), PL(REG)$ *are incomparable, excepting the case of* $ARB(REG), PL(REG)$ *for which we have the proper inclusion* $ARB(REG) \subset PL(REG)$.

(ii) *Each pair* $(PL(FIN), PR(FIN))$, $(PL(FIN), LM(FIN))$, $(PR(FIN), LM(FIN))$, $(ARB(FIN), PR(FIN))$, $(ARB(FIN), LM(FIN))$ *consists of incomparable families; the following inclusions are proper:* $GL(FIN) \subset X(FIN)$ *for all* $X \in \{ARB, PL, PR, LM\}$, *and* $ARB(FIN) \subset PL(FIN)$.

Lemma 1. *For each $X \in \{ARB, PR, LM, GL\}$ we have $PL(FIN) - X(REG) \neq \emptyset$.*

Proof. Follows from the fact that $PL(FIN)$ contains one-letter non-regular languages, but the one-letter languages in the other families F are regular (Theorem 3). $\Diamond$

Lemma 2. $ARB(F) \subseteq PL(F), F \in \{FIN, REG\}$.

Proof. For a shuffle grammar $G = (V, B, (R_1, C_1), \ldots, (R_n, C_n))$ used in the arbitrary mode of derivation, construct $G' = (V, B, (R_1, C_1), \ldots, (R_n, C_n), (V, \{\lambda\}))$.

For $a_1a_2 \ldots a_kx_2b_1b_2 \ldots b_j \Longrightarrow_i^{arb} a_1a_2 \ldots a_kx_2'b_1b_2 \ldots b_j$ in G, with $x_2 \in R_i, x_2' \in x_2 ⧢ C_i$, for some $1 \leq i \leq n$, we have $a_s = a_s ⧢ \lambda, 1 \leq s \leq k$, $b_s = b_s ⧢ \lambda, a \leq s \leq j$, hence $a_1a_2 \ldots a_kx_2b_1b_2 \ldots b_j \Longrightarrow^{pl} a_1a_2 \ldots a_kx_2'b_1b_2 \ldots b_j$ in G'.

Conversely, a derivation $x_1 \ldots x_k \Longrightarrow^{pl} x_1' \ldots x_k'$ in G' corresponds to a derivation $x_1x_2 \ldots x_k \Longrightarrow_{i_1}^{arb} x_1'x_2 \ldots x_k \Longrightarrow_{i_2}^{arb} x_1'x_2'x_3 \ldots x_k \Longrightarrow_{i_3}^{arb} \ldots \Longrightarrow_{i_k}^{arb} x_1' \ldots x_k'$ in G, with the steps $\Longrightarrow_{i_j}^{arb}$ omitted when $x_j' = x_j$.

In conclusion, $L_{arb}(G) = L_{pl}(G')$. $\Diamond$

Lemma 3. *The language $L_{gl}(G)$, for $G = (\{a, b\}, \{ab\}, (a^+b^+, \{ab\}))$ is not in $ARB(REG)$, $PR(REG)$, or $LM(REG)$.*

Proof. We have

$$\begin{aligned} L_{gl}(G) = \{a^nb^n \mid n \geq 1\} \cup \\ \cup \{a^naba^mb^{n+m} \mid n, m \geq 0\} \cup \\ \cup \{a^{n+m}b^nabb^m \mid n, m \geq 0\}. \end{aligned}$$

Assume that this language can be generated in one of the modes arb, pr, lm by a grammar $G' = (\{a, b\}, B, (R_1, C_1), \ldots, (R_n, C_n))$. Each string in $L_{gl}(G)$ contains the same number of occurrences of a and of b, hence for each string in C_i which is used in a derivation we must have the same property. In the mentioned modes f of derivation, if we have a derivation $x_1x_2x_3 \Longrightarrow_i^f x_1x_2'x_3$ (for $f = pr$ we have $x_1 = \lambda$) using a string $u \in C_i$, then we can obtain $x_1x_2x_3 \Longrightarrow_i^f x_1x_2ux_3$, too, hence the use of u can be iterated, thus producing strings $x_1x_2u^nx_3$ with arbitrary n. For $u \neq \lambda$ this is contradictory ($|u|_a = |u|_b$). $\Diamond$

Lemma 4. *The language $L = ab^+a$ is in $ARB(FIN) \cap LM(FIN)$ but not in $GL(REG) \cup PR(REG)$.*

Proof. The language L can be generated by the grammar $G = (\{a, b\}, \{aba\}, (\{b\}, \{b\}))$ in both modes of derivation arb and lm, but it cannot be generated by any shuffle grammar in modes gl or pr: in order to arbitrarily increase the number of b occurrences we have to use a string $b^r, r \geq 1$, which is shuffled to a string containing the leftmost occurrence of the symbol a in the strings in L. In this way, strings beginning with b^ra can be produced, a contradiction. $\Diamond$

Lemma 5. *The language $L = ab^+a \cup bab^+a$ is in $PR(FIN)$ but not in $GL(REG)$.*

Proof. For $G = (\{a, b\}, \{aba\}, (\{ab\}, \{b\}))$ we have $L_{pr}(G) = L$ but L cannot be generated in the gl mode by any grammar: we need a string u which introduces occurrences of b and in the global mode this u can be adjoined in the right hand of the rightmost occurrence of a in the strings of L. $\Diamond$

Lemma 6. *The language $L_{pr}(G)$, for the grammar $G = (\{a, b\}, \{ab\}, (\{ab\}, \{ab\}))$, is not in $LM(REG)$.*

Proof. We have

$$L_{pr}(G) = (ab)^+ \cup aabb(ab)^*.$$

(When the leftmost two symbols are not ab, the derivation is blocked.)

Assume that $L_{pr}(G) = L_{lm}(G')$ for some $G' = (\{a, b\}, B, (R_1, C_1), \ldots, (R_n, C_n))$. For every $z \in L_{pr}(G)$ we have $|z|_a = |z|_b$, hence the same property holds for all strings in sets C_i effectively used in derivations. Take such a nonempty string $u \in C_i$ and examine the form of strings $x \in R_i$. Such strings cannot be prefixes of $aabb(ab)^n, abb(ab)^n, bb(ab)^n$, because otherwise we can obtain parasitic strings by introducing u in the left of the pair bb in strings $aabb(ab)^m$. Suppose that x is a prefix of $b(ab)^n$ different from b. Then we can shuffle the string u in such a way to obtain a pair aa in the right of the pair bb already existing in strings of the form $aabb(ab)^n$. Again a contradiction, hence the strings of the form $aabb(ab)^n$ cannot be derived. This implies that they are obtained by derivations of the form $(ab)^n \Longrightarrow_i^{lm} aabb(ab)^m$. Consequently, also $aabb(ab)^n \Longrightarrow^{lm} aabbaabb(ab)^m$ is possible (we have found that the prefix $aabb$ cannot be rewritten, hence the derivation is leftmost). Such strings are not in $L_{pr}(G)$. ♢

Lemma 7. *The language $L = L_{pr}(G)$ in the previous lemma is not in the family $PL(REG)$.*

Proof. Assume that $L = L_{pl}(G')$ for some $G' = (\{a,b\}, B, (R_1, C_1), \ldots, (R_n, C_n))$.

If a derivation $(ab)^n \Longrightarrow^{pl} aabb(ab)^m$ is possible in G', then also $(ab)^n(ab)^n \Longrightarrow^{pl} aabb(ab)^m aabb(ab)^m$ is possible, a contradiction. Consequently, the strings $aabb(ab)^q$ can be produced only by derivations of the form $aabb(ab)^p \Longrightarrow^{pl} aabb(ab)^q$.

Assume that there is a derivation of the form $(ab)^n \Longrightarrow^{pl} (ab)^m$.

We cannot have pairs (R_i, C_i) with $\lambda \in R_i, C_i \neq \{\lambda\}$. Indeed, if such a pair exists, then from a derivation $aabb(ab)^p \Longrightarrow^{pl} aabb(ab)^q$ we can produce also $uaabb(ab)^q$, for each $u \in C_i$.

Examine the pairs $(x,u), x \in R_i, u \in C_i$, used in the derivation $(ab)^n \Longrightarrow^{pl} (ab)^m$. We know that $x \neq \lambda$. If x contains a symbol a and also u contains an occurrence of a, then we can produce strings having the subword aa; this is forbidden ($(ab)^n$ cannot generate strings of a form different from $(ab)^r$). Similarly, we cannot have occurrences of b both in x and in u. Therefore, $x = a^j, u = b^k$ or $x = b^s, u = a^t$. Clearly, we must have $i = s = 1$ (no other subwords consisting of a or of b only appear in the derived string) and $j = t = 1$ (no other subwords consisting of a or of b only appear in the produced string). Such pairs (x,u) can be clearly used for producing a parasitic string, a contradiction.

In conclusion, the strings $(ab)^n$ cannot be derived, hence such strings with arbitrarily long n must be obtained by derivations $aabb(ab)^p \Longrightarrow^{pl} (ab)^n$. Then the symbols aa and bb in the prefix of $aabb(ab)^p$ must be separated by an occurrence of b, respectively of a. Irrespective whether this a must be introduced *after* the first b or *before* the second b, we can introduce it *before* the first b, respectively *after* the second b, thus preserving the pair bb. If the pair aa in its left-hand has been correctly shuffled with b, then we obtain a string of the form x_1bbx_2 with $|x_1| \geq 3$, which is not in L. The equality $L = L_{pl}(G')$ is impossible. ♢

Lemma 8. *There are languages in $GL(REG) - PL(REG)$.*

Proof. Consider the grammar

$$G = (\{a,b\}, \{ab\}, ((ab)^*, \{ab\})).$$

We have

$$L_{gl}(G) = (ab)^+ \cup (ab)^* a(ab)^+ b(ab)^*,$$

which is not a parallel shuffle language. The argument is similar to that in the previous proof, hence it is left to the reader. ♢

Lemma 9. *There are languages in $ARB(FIN) - LM(REG)$.*

Proof. For the grammar $G = (\{a,b\}, \{abba\}, (\{a\}, \{a\}))$ we have

$$L_{arb}(G) = a^+bba^+.$$

Assume that $L_{arb}(G) = L_{lm}(G')$ for some $G' = (\{a,b\}, B, (R_1, C_1), \ldots, (R_n, C_n))$. For every $u \in C_i$ effectively used, we must have $u = a^r, r \geq 0$. No derivation can use a pair $(x,u), x \in R_i, u \in C_i$, with x containing a symbol b, otherwise we can produce strings of the form $a^n ba^p ba^m, p \geq 1$. Consequently, all selectors R_i can be supposed to contain only strings in a^*. A pair (R_i, C_i) can be used in a derivation if (and only if) the pair $(\{x_i\}, C_i)$ can be used, for x_i the shortest string in R_i; for two pairs $(\{x_i\}, C_i), (\{x_j\}, C_j)$, only that with the shortest x_i, x_j can be used. Denote

$$k = \min\{|x| \mid x \in R_i, 1 \leq i \leq n\}.$$

From a string $a^n bba^m$ in B with $n < k, m < k$ we can produce no other string; when $n \geq k$, we produce only strings of the form $a^t bba^m$ without modifying the number m; when $n < k$ and $m \geq k$, then we can produce strings of the form $a^n bba^t$. However, $L_{arb}(G)$ contains strings $a^s bba^t$ with simultaneously arbitrarily large s, t, a contradiction. $\diamondsuit$

Lemma 10. *The language generated by $G = (\{a, b\}, \{ab\}, (\{ab\}, \{ab\}))$ in the leftmost mode is not in $PL(REG)$.*

Proof. We have

$$L_{lm}(G) = (ab)^+ \cup \{a^i((ab)^*b)^i(ab)^* \mid i \geq 1\}.$$

Here is a derivation in G in the leftmost mode:

$$\begin{aligned} ab &\Longrightarrow^{lm} abab \Longrightarrow^{lm} ababab \Longrightarrow^{lm} \ldots \Longrightarrow^{lm} (ab)^n \Longrightarrow^{lm} \\ &\Longrightarrow^{lm} aabb(ab)^{n-1} \Longrightarrow^{lm} \ldots \Longrightarrow^{lm} a(ab)^m b(ab)^{n-1} \Longrightarrow^{lm} \\ &\Longrightarrow^{lm} aaabb(ab)^{m-1}b(ab)^{n-1} \Longrightarrow^{lm} \ldots \Longrightarrow^{lm} aa(ab)^p b(ab)^{m-1}b(ab)^{n-1}. \end{aligned}$$

Assume that $L_{lm}(G) = L_{pl}(G')$ for some $G' = (\{a, b\}, B, (R_1, C_1), \ldots, (R_n, C_n))$.

Examine the possibilities to obtain the strings $(ab)^n$ with arbitrarily large n. We cannot have a derivation $w \Longrightarrow^{pl} (ab)^n$ for w a string in $a^i((ab)^*b)^i(ab)^*, 1 \geq 1$, because we must separate both the i left occurrences of a and the i pairs bb appearing in the string; irrespective of the used strings to be shuffled we can do only part of these operations, letting for instance a pair aa unchanged and separating all pairs bb; we obtain a parasitic string. Therefore we must have derivations $(ab)^n \Longrightarrow^{pl} (ab)^m$, $m > n$. If in such a derivation we use a pair $(x, u), x \in R_i, u \in C_i$, such that a occurs both in x and in u, then we can produce strings with substrings aa. Starting from a string $(ab)^n(ab)^n$ we can then produce strings having aa not in the left-hand end, a contradiction. It follows that symbols a are introduced only by pairs (x, u) with $x = b$ (no other subword of $(ab)^n$ consists of only occurrences of b). In order to cover $(ab)^n$ we must use also pairs (y, v) with y containing a and v not containing a. This implies $v = b$ or $v = \lambda$. The use of (b, u) introduces at least one new occurrence of a for each b, hence, in order to keep the number of a and b equal, we must have $v = b$, which implies $y = a$. Using such pairs we can produce again strings containing pairs aa not in the left-hand end, a contradiction which concludes the proof. $\diamondsuit$

Let us note that in all previous lemmas the considered languages are generated by grammars with only one component (R, C). Consequently, we have

Corollary 1. *For all families $X(F), X'(F)$ with $X, X' \in \{ARB, PR, LM, GL, PL\}$, $X \neq X'$, $F \in \{FIN, REG\}$, which are incomparable, all $X_i(F), X'_j(F)$ are incomparable, too, for every $i, j \geq 1$.*

Corollary 2. *Each family F such that $REG \subseteq F \subseteq CF$ is incomparable with each family $ARB(REG), PR(REG), LM(REG), GL(REG), PL($*

Proof. The fact that $X(REG) - CF \neq \emptyset$, for each $X \in \{ARB, PR, LM, GL, PL\}$, has been already pointed out. On the other hand, the language ab^+a in Lemma 4 is not in $GL(REG)$ or in $PR(REG)$, and the language $(ab)^+ \cup aabb(ab)^*$ in Lemma 6 is not in $LM(REG)$ or on $PL(REG)$, hence it is not in $ARB(REG)$. These languages are regular, which concludes the proof. $\diamondsuit$

By a straightforward simulation in terms of context-sensitive grammars of checking the conditions defining the correct derivation in a shuffle grammar and of performing such a derivation, we get

Theorem 8. *All families $X(F), X \in \{ARB, PR, LM, GL, PL\}, F \in \{FIN, REG\}$, are strictly included in CS.*

The properness of these inclusions follows from the previous Corollary 2.

4 The case of using maximal selectors

For a shuffle grammar $G = (V, B, (R_1, C_1), \ldots, (R_n, C_n))$ as in the previous sections, we can also define the following mode of derivation: for $x, y \in V^*$ and $1 \leq i \leq n$, write

$$\begin{aligned} x \Longrightarrow_i^{max} \ iff\ & x = x_1x_2x_3, x_1, x_2 \in V^*, x_2 \in R_i, \\ & y = x_1x_2'x_3,\ for\ some\ x_2' \in x_2 \sqcup\!\sqcup u, u \in C_i, \\ & and \\ & there\ is\ no\ decomposition\ of\ x\ of\ the\ form \\ & x = x_1'x_1''x_2x_3'x_3'',\ with\ x_1''x_3' \neq \lambda, \\ & x_1''x_2x_3' \in R_j\ for\ some\ j, 1 \leq j \leq n. \end{aligned}$$

(We use x_2 for shuffling only if no longer string can be used containing that occurrence of x_2 as a substring.)

We denote by $L_{max}(G)$ the language generated in this way and by $MAX(FIN)$, $MAX(REG)$ the corresponding families of languages.

Lemma 11. $MAX_1(FIN) - X(REG) \neq \emptyset, X \in \{PR, LM, GL\},\ MAX_2(FIN) - ARB(REG) \neq \emptyset.$

Proof. For $X \in \{PR, GL\}$ the assertion is proved by the language in Lemma 4: in general, if $G = (V, B, (\{x\}, C))$ (one component, with a singleton selector), then the maximality has no effect, the maximal derivations in G are arbitrary derivations. This is the case with the grammar in Lemma 4.

This is also true for the grammar in the proof of Lemma 9; that language covers the case $X = LM$.

Consider now the grammar

$$G = (\{a, b, c\}, \{cab\}, (\{cab\}, \{ab\}), (\{acab, bcab\}, \{\lambda\}).$$

Starting from a string $c(ab)^n$ (initially we have $n = 1$), we can shuffle the string ab in the prefix ab in five essentially different ways:

$$\begin{aligned} &(1)\ c(ab)^n \Longrightarrow^{max} abc(ab)^n, \\ &(2)\ c(ab)^n \Longrightarrow^{max} acb(ab)^n, \\ &(3)\ c(ab)^n \Longrightarrow^{max} acabb(ab)^{n-1}, \\ &(4)\ c(ab)^n \Longrightarrow^{max} c(ab)^{n+1}, \\ &(5)\ c(ab)^n \Longrightarrow^{max} caabb(ab)^{n-1}. \end{aligned}$$

In cases (2) and (5) the derivation cannot continue (all selectors contain the subword cab); in cases (1) and (3) we cannot use the selector cab because $acab$ or $bcab$ are present; the latter selectors entail the shuffle of λ, which changes nothing, hence the derivation is blocked. Only case (4) can be continued. Consequently,

$$L_{max}(G) = c(ab)^+ \cup abc(ab)^+ \cup acb(ab)^+ \cup acabb(ab)^* \cup caabb(ab)^*.$$

This language cannot be generated by a shuffle grammar $G' = (\{a, b, c\}, B, (R_1, C_1), \ldots, (R_n, C_n))$ in the arbitrary mode. Assume the contrary. Every string in B, in a grammar G' as above, must contain the symbol c and every string in $C_i, 1 \leq i \leq n$, must contain the same number of occurrences of a and of b. If $\lambda \in C_i$, then this string can be ignored without modifying the language of G'. In order to generate strings $c(ab)^m$ with arbitrarily large m we must have derivation steps $w \Longrightarrow^{arb} c(ab)^m$ with $w = c(ab)^p$ or $w = caabb(ab)^p, p < m$ (the other strings in $L_{max}(G)$ contain symbols a or b in front of c). Starting from $caabb(ab)^p$ we have to introduce one occurrence of b between symbols a in the substring aa and one occurrence of a between symbols b in bb, that is we need a pair (z, u) with $z \in R_i, u \in C_i$, $z = z_1abz_2$ and $u = u_1bu_2au_3$. Then from $caabb(ab)^p$ we can also produce $x_1z_1u_1bu_2au_3abz_2x_2 = x_1z_1u_1bu_2au_3abbx_2'$, a parasitic string. If $c(ab)^p \Longrightarrow^{arb} c(ab)^m$ uses a pair (z, u) with $z \in c\{a, b\}^*$, then $u^sc(ab)^p$ can be produced, for arbitrary $s \geq 1$ (we introduce u in the left of c, thus leaving the occurrence of z in $c(ab)^p$ unchanged). Consequently, we have to use a pair (z, u) with $z \in \{a, b\}^*$. More exactly, z is a substring of $(ab)^p$. As u contains occurrences of both a and b, we can derive $c(ab)^s(ab)^p$ with arbitrary s (all such strings are in $L_{max}(G)$) in such a way to obtain $c(ab)^s y$ with y containing a substring aa or a substring bb. Such a string is not in $L_{max}(G)$, hence the grammar G' cannot generate strings $c(ab)^m$ with arbitrarily large m. The equality $L_{max}(G) = L_{arb}(G')$ is impossible. $\diamondsuit$

Lemma 12. $GL(F) \subseteq MAX(F), F \in \{FIN, REG\}$.

Proof. As $GL(FIN) = FIN$ and, clearly, $FIN \subseteq MAX(FIN)$, the case $F = FIN$ is obvious.

Take now a grammar $G = (V, B, (R_1, C_1), \ldots, (R_n, C_n))$ with regular $R_i, 1 \leq i \leq n$, and construct $G' = (V, B, (R_1, C_1), \ldots, (R_n, C_n), (R_{n+1}, C_{n+1}))$ with

$$R_{n+1} = V^*(\bigcup_{i=1}^{n} R_i)V^*, \quad C_{n+1} = \{\lambda\}.$$

We have $L_{gl}(G) = L_{max}(G')$.

($\subseteq$) If $x \Longrightarrow_i^{gl} y$, that is $x \in R_i, y \in x \sqcup\!\sqcup u, u \in C_i$, then the derivation is maximal, we also have $x \Longrightarrow_i^{max} y$. Consequently, if $w \in B$ derives z in G in the global mode, then w derives z in G' in the maximal mode.

($\supseteq$) Take a derivation in G', $x \Longrightarrow_i^{max} y$. If $i = n+1$, then $x = y$. If $i \leq n$, then we have $x \in R_i$, otherwise the derivation is not allowed: for $x = x_1x_2x_3, x_2 \in R_i, x_1x_2 \neq \lambda$, we have $x_1x_2x_3 \in R_{n+1}$, hence the derivation in (R_i, C_i) is not allowed. By induction on the length of derivations, we can now show that for every maximal derivation in G' there is a global derivation in G producing the same string, hence $L_{max}(G') \subseteq L_{gl}(G)$. $\diamondsuit$

Lemma 13. $PR(FIN) - MAX(REG) \neq \emptyset$.

Proof. Consider the grammar

$$G = (\{a, b\}, \{ab\}, (\{ab\}, \{ab\})).$$

We obtain

$$L_{pr}(G) = (ab)^+ \cup aabb(ab)^*.$$

This language is not in $MAX(REG)$. Assume the contrary, that is $L_{pr}(G) = L_{max}(G')$ for some $G' = (\{a, b\}, B, (R_1, C_1), \ldots, (R_n, C_n))$.

If we have a derivation of the form $aabb(ab)^n \Longrightarrow^{max} (ab)^m$, then we have to separate the symbols in the subwords aa and bb by b or some $bxb, x \in \{a, b\}^*$, and by a or some $aya, y \in \{a, b\}^*$, respectively. To this aim, a string of the form z_1abz_2 must be used as selector and a string $u = u_1bu_2au_3$ must be shuffled. But then also $aabb(ab)^n \Longrightarrow^{max} au_1bu_2au_3abb(ab)^n$ is possible, and this is not in the language $L_{pr}(G)$, a contradiction.

Assume that we have a derivation $(ab)^n \Longrightarrow^{max} (ab)^m, m > n \geq 2$. Then we must have $(ab)^n = x_1x_2x_3, x_2 \in R_i$, for some i such that $(ab)^m = x_1x_2'x_3, x_2' \in x_2 \sqcup\!\sqcup v, v \in C_i$. Clearly, $|v|_a = |v|_b \geq 1$. Then from $(ab)^{2n}$ we can derive $(ab)^n(ab)^m$. If $x_2 = \lambda$ and v starts by an occurrence of $a, v = av'$, then we can produce $(ab)^n z_1aav'bz_2$, for some $z_1, z_2 \in \{a, b\}^*$; if v starts by $b, v = bv'$, then we can produce $(ab)^n z_1abbv'z_2$, for some $z_1, z_2 \in \{a, b\}^*$. No one of these strings is in $L_{pr}(G)$, a contradiction. Therefore we must have $x_2 \neq \lambda$. If x_2 contains the symbol a, then we can obtain $(ab)^n z_1aaz_2$, for some $z_1, z_2 \in \{a, b\}^*$, by appropriate shuffling of x_2 and v. If x_2 contains the letter b, then we can obtain $(ab)^n z_1bbz_2$, for some $z_1, z_2 \in \{a, b\}^*$, by appropriate shuffling. In both cases we have obtained parasitic strings.

Consequently, the strings $(ab)^n$ cannot be generated, they must be introduced as axioms; this is impossible, because the set B is finite, hence $L_{pr}(G) \notin MAX(REG)$. $\diamondsuit$

Lemma 14. *A one-letter language is regular if and only if it is in $MAX(FIN)$.*

Proof. The proof of Theorem 3 shows the fact that every one-letter regular language is in $MAX(FIN)$.

Conversely, take a grammar $G = (\{a\}, B, (R_1, C_1), \ldots, (R_n, C_n))$ and write each R_i, $1 \le i \le n$, in the form

$$R_i = H_i \cup \bigcup_{s=1}^{k_i} \{a^m \mid m = p_{i,s} + j \cdot q_i, j \ge 0\},$$

for H_i finite languages, and q_i associated with $R_i, 1 \le i \le n$. Denote

$$T = \max\{q_i \mid 1 \le i \le n\},$$
$$K = \max\{t \mid a^t \in B \ or \ a^t \in H_i, \ or \ t = p_{i,s}, 1 \le s \le k_i, 1 \le i \le n\}.$$

Take for each R_i a deterministic finite automaton

$$A_i = (Q_i, \{a\}, s_{0,i}, F_i, \delta_i), 1 \le i \le n.$$

We construct the right-linear grammar $G' = (N, \{a\}, S, P)$ with

$$N = Q_1^T \times Q_2^T \times \ldots \times Q_n^T \cup \{S\},$$

and P contains the following rules:

(1) $S \to a^r$, for $a^r \in L(G), r \le K + T$;
(2) $S \to a^r(s_{1,1}s_{2,1} \ldots s_{T,1}, s_{1,2}s_{2,2} \ldots s_{T,2}, \ldots, s_{1,n}s_{2,n} \ldots s_{T,n})$,
for $K + T \le r < K + 2T, \delta_i(s_{0,i}, a^{r-T+j}) = s_{j,i}, 1 \le j \le T, 1 \le i \le n$,
(3) $(s_{1,1}s_{2,1} \ldots s_{T,1}, \ldots, s_{1,n}s_{2,n} \ldots s_{T,n}) \to a^r(s'_{1,1}s'_{2,1} \ldots s'_{T,1}, \ldots, s'_{1,n}s'_{2,n} \ldots s'_{T,n})$,
where $a^r \in C_i$ for some $i \in \{1, 2, \ldots, n\}$ such that $s_{T-j,i} \in F_i$ and $s_{m,l} \in Q_l - F_l$ for all $1 \le l \le n$ and $T - j + 1 \le m \le T$; moreover, $s'_{j,l} = \delta_l(s_{j,l}, a^r), 1 \le j \le T, 1 \le l \le n$.
(4) $(s_{1,1}s_{2,1} \ldots s_{T,1}, \ldots, s_{1,n}s_{2,n} \ldots s_{T,n}) \to \lambda$ for all $(s_{1,1}s_{2,1} \ldots s_{T,1}, \ldots, s_{1,n}s_{2,n} \ldots s_{T,n}) \in N$.

For every language R_i, the difference between the length of two consecutive strings in R_i is bounded by q_i. Therefore the difference between two strings in any of these languages is bounded by T. In the nonterminals of G' we memorize the last T states used by the automata A_i when recognizing the current string. A prolongation to right is possible (by rules of type (3)) only according to that language R_i which contains the largest subword of the current string (thus we check the maximality). The derivation can stop in any moment by rules of type (4). Consequently, $L_{max}(G) = L(G')$, hence $L_{max}(G) \in REG$, which concludes the proof. $\diamondsuit$

Summarizing these lemmas and the fact that $PL(FIN)$ contains one-letter non-regular languages, we obtain

Theorem 9. *$MAX(F)$ includes strictly $GL(F)$ and is incomparable with $PR(F')$, $F, F' \in \{FIN, REG\}$; $MAX(FIN) - X(REG) \neq \emptyset$ for $X \in \{ARB, PR, GL, LM\}$.*

Open problems. Are the differences $ARB(F) - MAX(F), LM(F) - MAX(F)$, $MAX(F) - PL(F)$, for $F \in \{FIN, REG\}$, non-empty ?

The diagram in figure 1 summarizes the results in the previous sections (an arrow from X_1 to X_2 indicates the strict inclusion of the family X_1 in X_2.)

The maximal derivation discussed above can be considered as arbitrary-maximal, since we are not concerned with the place of the selector in the rewritten string, but only with its maximality. Similarly, we can consider prefix-maximal and leftmost-maximal derivations, when a maximal prefix or a substring which is both maximal and leftmost is used for derivation, respectively. We hope to return to these cases. At least the prefix-maximal case looks quite interesting. For instance, if the sets R_i are disjoint, then in every step of a derivation only one of them can be used, which decreases the degree of nondeterminism of the derivations.

5 Final remarks

We have not investigated here a series of issues which are natural for every new considered generative mechanisms, such as closure and decidability problems, syntactic complexity, or the relations with other grammars which do not use the rewriting in generating languages. Another important question is whether or not the degree of shuffle grammars induces an infinite hierarchy of languages. We close this discussion by emphasizing the variety of places where the shuffle operation proves to be useful and interesting, as well as the richness of the area of grammars based on adjoining strings.

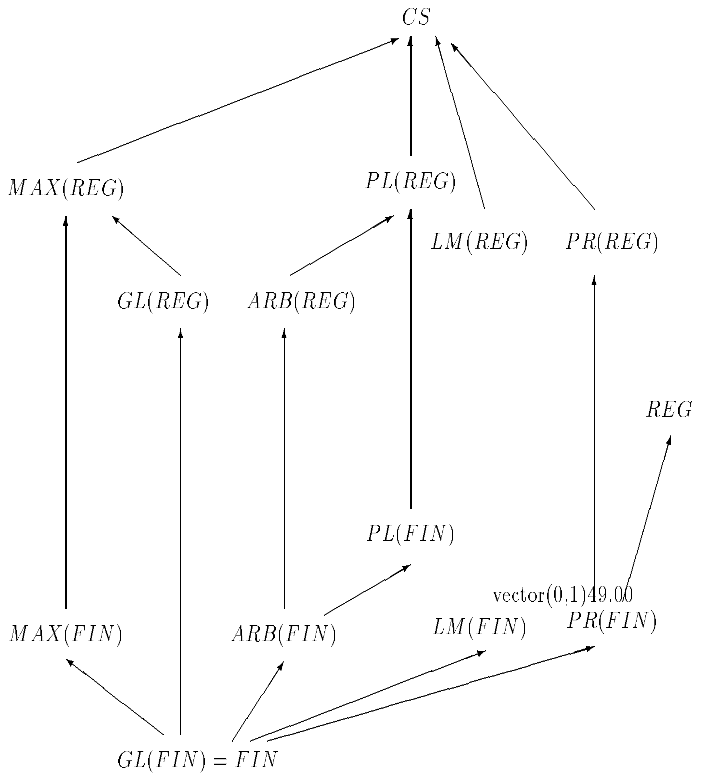

Fig. 1

References

[1] B. Berard, Literal shuffle, *Theoretical Computer Science*, 51 (1987), 281 – 299.

[2] J. Dassow, Gh. Paun, A. Salomaa, Grammars based on patterns, *Intern. J. Found. Computer Sci.*, 4, 1 (1993), 1 – 14.

[3] B. C. Galiukschov, Polukontekstnie gramatiki, *Mat. Logica i Mat. Ling.*, Kalinin Univ, 1981, 38 – 50 (in Russ.).

[4] G. H. Higman, Ordering by divisibility in abstract algebra, *Proc. London Math. Soc.*, 3 (1952), 326 – 336.

[5] M. Ito, G. Thierrin, S. S. Yu, Shuffle-closed languages, submitted, 1993.

[6] M. Jantzen, On shuffle and iterated shuffle, *Actes de l'ecole de printemps de theorie des langages* (M. Blab, Ed.), Murol, 1981, 216 – 235.

[7] A. K. Joshi, S. R. Kosaraju, H. M. Yamada, String adjunct grammars: I. Local and distributed adjunction, *Inform. Control*, 21 (1972) , 93 – 116.

[8] S. Marcus, Contextual grammars, *Rev. Roum. Math. Pures Appl.*, 14, 10 (1969), 1525 – 1534.

[9] Al. Mateescu, Marcus contextual grammars with shuffled contexts, in *Mathematical Aspects of Natural and Formal Languages* (Gh. Păun, ed.), World Sci. Publ., Singapore, 1994, 275 – 284.

[10] L. Nebesky, The ξ-grammar, *Prague Studied in Math. Ling.*, 2 (1966), 147 – 154.

[11] Gh. Paun, *Grammars for Economic Processes*, The Technical Publ. House, Bucuresti, 1980 (in Romanian).

[12] Gh. Paun, *Contextual Grammars*, The Publ. House of the Romanian Academy of Sciences, Bucureşti, 1982 (in Romanian).

[13] Gh. Păun, G. Rozenberg, A. Salomaa, Contextual grammars: Erasing, determinism, one-sided contexts, in *Developments in Language Theory* (G. Rozenberg, A. Salomaa, eds.), World Sci. Publ., Singapore, 1994, 370 – 388.

[14] Gh. Paun, G. Rozenberg, A. Salomaa, Contextual grammars: Parallelism and blocking of derivation, *Fundamenta Informaticae*, to appear.

[15] Gh. Paun, G. Rozenberg, A. Salomaa, Marcus contextual grammars: Modularity and leftmost derivation, in *Mathematical Aspects of Natural and Formal Languages* (Gh. Paun, ed.), World Sci. Publ., Singapore, 1994, 375 – 392.

[16] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, London, 1980.

[17] A. Salomaa, *Formal Languages*, Academic Press, New York, London, 1973.