Journal of Universal Computer Science, vol. 18, no. 14 (2012), 2024-2040
submitted: 14/1/12, accepted: 16/4/12, appeared: 28/7/12 © J.UCS

Controlled Pure Grammar Systems

Alexander Meduna
(Brno University of Technology, Faculty of Information Technology
IT4Innovations Centre of Excellence, Brno, Czech Republic
meduna@fit.vutbr.cz)

Petr Zemek
(Brno University of Technology, Faculty of Information Technology
IT4Innovations Centre of Excellence, Brno, Czech Republic
izemek@fit.vutbr.cz)

Abstract: This paper discusses grammar systems that have only terminals, work in the leftmost way, and generate their languages under the regulation by control languages over rule labels. It establishes three results concerning their generative power. First, without any control languages, these systems are not even able to generate all context-free languages. Second, with regular control languages, these systems, having no more than two components, characterize the family of recursively enumerable languages. Finally, with control languages that are themselves generated by regular-controlled context-free grammars, these systems over unary alphabets generate nothing but regular languages. In its introductory section, the paper gives a motivation for introducing these systems, and in the concluding section, it formulates several open problems.

Key Words: formal languages, pure grammar systems, controlled derivations
Category: F.4.2, F.4.3, F.1.1

1 Introduction

Indisputably, context-free grammars are central to formal language theory as a whole (see [Rozenberg and Salomaa 1997, Dassow and Păun 1989, Meduna 2000, Salomaa 1973]). It thus comes as no surprise that this theory has introduced a broad variety of their modified versions, ranging from simplified and restricted versions up to fundamentally generalized systems based upon these grammars. Grammar systems (see [Csuhaj-Varjú et al. 1994]), regulated context-free grammars (see [Dassow and Păun 1989, Martín-Vide et al. 2004]), pure context-free grammars (see [Maurer et al. 1980, Mäkinen 1986, Martinek 1998] and page 242 in [Rozenberg and Salomaa 1997]), and context-free grammars with leftmost derivations (see [Meduna 2007] and Section 5.1 in [Meduna 2000]) definitely belong to the key modifications of this kind. Next, we give an insight into these four modifications.

I. Grammar systems consist of several context-free grammars, referred to as their components, which mutually cooperate and, in this way, generate the languages of these systems.

II. Regulated context-free grammars prescribe the use of rules during derivations by some additional regulating mechanisms, such as control languages over the label of grammatical rules.

III. Pure context-free grammars simplify ordinary context-free grammars by using only one type of symbols—terminals. There exist pure sequential versions of context-free grammars as well as pure parallel versions of context-free grammars, better known as 0L systems (see [Rozenberg and Salomaa 1980, Rozenberg and Salomaa 1986, Rozenberg and Salomaa 1997]).

IV. Context-free grammars that perform only leftmost derivations fulfill a key role in an principal application area of these grammars—parsing (indeed, see [Meduna 2007, Aho et al. 2006, von zur Gathen and Gerhard 2003]).

Of course, formal language theory has also investigated various combinations of I through IV. For instance, combining I and III, pure grammar systems have been studied (see [Bensch and Bordihn 2007], [Bordihn et al. 1999], and [Aydin and Bordihn 2003]). Similarly, based upon various combinations of I and II, a number of regulated grammar systems were defined and discussed (see [Beek and Kleijn 2002, Fernau and Holzer 2002, Csuhaj-Varjú et al. 1993, Lukáš and Meduna 2006, Păun 1993, Goldefus 2009, Lukáš and Meduna 2010] and [Csuhaj-Varjú and Vaszil 2001]). Following this vivid investigation trend, the present paper combines all four modifications mentioned above.

More specifically, this paper introduces pure grammar systems that generate their languages in the leftmost way, and in addition, this generative process is regulated by control languages over rule labels. The paper concentrates its attention on investigating the generative power of these systems. It establishes three major results. First, without any control languages, these systems are not even able to generate all context-free languages (Theorem 11). Second, with regular control languages, these systems characterize the family of recursively enumerable languages, and this result holds even if these systems have no more than two components (Theorems 13 and 14). Finally, this paper considers control languages as languages that are themselves generated by regular-controlled context-free grammars; surprisingly enough, with control languages of this kind, these systems over unary alphabets generate nothing but regular languages (Theorem 15).

The paper is organized as follows. First, Sections 2 and 3 give all the necessary terminology. Then, Section 4 rigorously establishes the results mentioned above. In the conclusion, Section 5 formulates several open problems.

2 Preliminaries

In this paper, we assume that the reader is familiar with formal language theory (see [Dassow and Păun 1989, Meduna 2000, Rozenberg and Salomaa 1997]). For a set P, $\mathrm{card}(P)$ denotes the cardinality of P. For a finite set of integers I, $\max(I)$ denotes the greatest integer in I. For an alphabet (finite nonempty set) V, V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation. If $\mathrm{card}(V) = 1$, then V is a *unary alphabet*. For $x \in V^*$, $|x|$ denotes the length of x, $\mathrm{rev}(x)$ denotes the reversal of x, and $\mathrm{alph}(x)$ denotes the set of symbols occurring in x. For $K \subseteq V^*$, we define $\mathrm{alph}(K) = \bigcup_{x \in K} \mathrm{alph}(x)$. If $\mathrm{card}(\mathrm{alph}(K)) = 1$, then K is a *unary language*.

A *finite automaton* is a quintuple, $M = (W, \Sigma, R, s, F)$, where W is a finite set of *states*, Σ is an *input alphabet*, $\Sigma \cap W = \emptyset$, $R \subseteq W \times (\Sigma \cup \{\varepsilon\}) \times W$ is a finite relation, $s \in W$ is the *start state*, and $F \subseteq Q$ is a set of *final states*. Members of R are referred to as *rules*, and instead of $(p, a, q) \in R$, we write $pa \rightarrow q$ throughout the paper. The *direct move relation* over $W\Sigma^*$, symbolically denoted by $\vdash$, is defined as follows: $pax \vdash qx$ in M if and only if $a \in \Sigma \cup \{\varepsilon\}$, $x \in \Sigma^*$, and $pa \rightarrow q \in R$. Let $\vdash^m$ and $\vdash^*$ denote the mth power of $\vdash$, for some $m \geq 1$, and the reflexive-transitive closure of $\vdash$, respectively. The *language accepted by* M is denoted by $L(M)$ and defined as $L(M) = \{w \in \Sigma^* \mid sw \vdash^* f$ with $f \in F\}$.

An *extended pushdown automaton* is a septuple, $M = (W, \Sigma, \Omega, R, s, \#, F)$, where W, Σ, s, and F are defined as in a finite automaton, Ω is an alphabet such that $\Sigma \subset \Omega$, $R \subseteq \Omega^* \times W \times (\Sigma \cup \{\varepsilon\}) \times \Omega^* \times W$ is a finite relation, and $\# \in \Omega - \Sigma$. The components Ω, R, and $\#$ are called the *pushdown alphabet*, the set of *rules*, and the *initial pushdown symbol*, respectively. By analogy with finite automata, instead of $(b, p, a, w, q) \in R$, we write $bpa \rightarrow wq$ throughout the paper. The *direct move relation* over $\Omega^* W \Sigma^*$, symbolically denoted by $\vdash$, is defined as follows: $ybpax \vdash ywqx$ in M if and only if $ybpax, ywqx \in \Omega^* W \Sigma^*$ and $bpa \rightarrow wq \in R$. Let $\vdash^m$ and $\vdash^*$ denote the mth power of $\vdash$, for some $m \geq 1$, and the reflexive-transitive closure of $\vdash$, respectively. The *language accepted by* M is denoted by $L(M)$ and defined as $L(M) = \{w \in \Sigma^* \mid \#sw \vdash^* f$ with $f \in F\}$.

A *context-free grammar* is a quintuple, $G = (N, T, \Psi, P, S)$, where N, T, and Ψ are three alphabets such that $N \cap T = \emptyset$, $S \in N$, and $P \subseteq \Psi \times N \times (N \cup T)^*$ is a finite relation such that if $(r, A, y), (s, A, y) \in P$, then $r = s$. Set $V = N \cup T$. The components V, N, T, Ψ, P, and S are called the *total alphabet*, the alphabet of *nonterminals*, the alphabet of *terminals*, the alphabet of *rule labels*, the set of *rules*, and the *start symbol*, respectively. Each $(r, A, y) \in P$ is written as $r\colon A \rightarrow y$ throughout this paper. The *direct derivation relation* over V^*, symbolically denoted by $\Rightarrow$, is defined as follows: $uAv \Rightarrow uyv\ [r]$ in G, or, simply, $uAv \Rightarrow uyv$

if and only if $u, v \in V^*$ and $r\colon A \to y \in P$. In the standard manner, we extend $\Rightarrow$ to $\Rightarrow^m$, where $m \geq 0$; then, based on $\Rightarrow^m$, we define $\Rightarrow^*$. The *language of* G is denoted by $L(G)$ and defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$.

Let $G = (N, T, \Psi, P, S)$ be a context-free grammar. If $r\colon A \to y \in P$ implies that $y \in T^*(N \cup \{\varepsilon\})$, then G is a *right-linear grammar*, and instead of $G = (N, T, \Psi, P, S)$ and $r\colon A \to y \in P$, we write just $G = (N, T, P, S)$ and $A \to y \in P$, respectively (i.e. we omit rule labels).

Let $G = (N, T, \Psi, P, S)$ be a context-free grammar, and let $\Xi \subseteq \Psi^*$ be a regular *control language*. The *language generated by* G *with* Ξ (see page 97 in [Dassow and Păun 1989] and page 251 in [Martín-Vide et al. 2004]) is denoted by $L(G, \Xi)$ and defined as $L(G, \Xi) = \{w \in T^* \mid S \Rightarrow^* w\ [\varrho] \text{ with } \varrho \in \Xi\}$. Less formally, $L(G, \Xi)$ consists of all strings over T for which there is a derivation from S according to a control string from Ξ. The pair (G, Ξ) is called a *regular-controlled grammar*.

The families of regular languages, context-free languages, and recursively enumerable languages are denoted by $\mathscr{L}(\text{REG})$, $\mathscr{L}(\text{CF})$, and $\mathscr{L}(\text{RE})$, respectively. Let $\mathscr{L}(\text{rC})$ denote the language family generated by regular-controlled grammars. Recall that right-linear grammars and finite automata characterize $\mathscr{L}(\text{REG})$, extended pushdown automata and context-free grammars characterize $\mathscr{L}(\text{CF})$, and recall that $\mathscr{L}(\text{REG}) \subset \mathscr{L}(\text{CF}) \subset \mathscr{L}(\text{rC}) \subset \mathscr{L}(\text{RE})$ (see [Martín-Vide et al. 2004]).

We will have a use of the following lemma.

Lemma 1. (See Theorem 1 in [Martín-Vide et al. 2004].) *All languages in* $\mathscr{L}(\text{rC})$ *over a unary alphabet are regular.*

Next, we slightly modify the notion of a queue grammar (introduced in [Kleijn and Rozenberg 1983]). A *left-extended queue grammar* (see [Meduna 2004, Kolář and Meduna 2000, Meduna 2003]) is a sextuple, $Q = (W, V, T, R, s, F)$, where W and V are two disjoint alphabets, $T \subseteq V$, $R \subseteq (V \times (W - F)) \times (V^* \times W)$ is a finite relation, $s \in (V - T)(W - F)$, and $F \subseteq W$. The components W, V, T, R, s, and F, are called the set of *states*, the *total alphabet*, the alphabet of *terminals*, the set of *rules*, the *start configuration*, and the set of *final states*, respectively. Assume that $\# \notin V \cup W$. If $u, v \in V^*\{\#\}V^*W$ so that $u = w\#axp$, $v = wa\#xyq$, $a \in V$, $x, y, w \in V^*$, $p, q \in W$, and $(a, p, y, q) \in R$, then $u \Rightarrow v\ [(a, p, y, q)]$ in G or, simply, $u \Rightarrow v$. In the standard manner, we extend $\Rightarrow$ to $\Rightarrow^m$, where $m \geq 0$; then, based on $\Rightarrow^m$, we define $\Rightarrow^*$. The *language of* Q is denoted by $L(Q)$ and defined as $L(Q) = \{z \in T^* \mid \#s \Rightarrow^* w\#zf \text{ for some } w \in V^* \text{ and } f \in F\}$.

Less formally, during every step of a derivation, a left-extended queue grammar shifts the rewritten symbol over $\#$; in this way, it records the derivation history, which represents a property fulfilling a crucial role in the proof of Lemma 12 in the next section.

Lemma 2. (See [Meduna 2004].) *For every recursively enumerable language K, there exists a left-extended queue grammar, $Q = (W, V, T, R, s, F)$, such that $L(Q) = K$, $T = \mathrm{alph}(K)$, $F = \{f\}$, $W = X \cup Y \cup \{\S\}$, where $X, Y, \{\S\}$ are pairwise disjoint, and every $(a, p, y, q) \in R$ satisfies either $a \in V - T$, $p \in X$, $y \in (V - T)^*$, $q \in X \cup \{\S\}$ or $a \in V - T$, $p \in Y \cup \{\S\}$, $y \in T^*$, $q \in Y$.*

Furthermore, Q generates every $h \in L(Q)$ in this way:

$$\begin{array}{ll}
\quad \#a_0p_0 & \\
\Rightarrow a_0\#x_0p_1 & [(a_0, p_0, z_0, p_1)] \\
\Rightarrow a_0a_1\#x_1p_2 & [(a_1, p_1, z_1, p_2)] \\
\vdots & \\
\Rightarrow a_0a_1 \cdots a_k\#x_kp_{k+1} & [(a_k, p_k, z_k, p_{k+1})] \\
\Rightarrow a_0a_1 \cdots a_ka_{k+1}\#x_{k+1}y_1p_{k+2} & [(a_{k+1}, p_{k+1}, y_1, p_{k+2})] \\
\vdots & \\
\Rightarrow a_0a_1 \cdots a_ka_{k+1} \cdots a_{k+m-1}\#x_{k+m-1}y_1 \cdots y_{m-1}p_{k+m} & [(a_{k+m-1}, p_{k+m-1}, \\
 & \; y_{m-1}, p_{k+m})] \\
\Rightarrow a_0a_1 \cdots a_ka_{k+1} \cdots a_{k+m}\#y_1 \cdots y_mp_{k+m+1} & [(a_{k+m}, p_{k+m}, y_m, \\
 & \; p_{k+m+1})]
\end{array}$$

where $k, m \geq 1$, $a_i \in V - T$ for $i = 0, \ldots, k + m$, $x_j \in (V - T)^*$ for $j = 1, \ldots, k + m$, $s = a_0p_0$, $a_jx_j = x_{j-1}z_j$ for $j = 1, \ldots, k$, $a_1 \cdots a_kx_{k+1} = z_0 \cdots z_k$, $a_{k+1} \cdots a_{k+m} = x_k$, $p_0, p_1, \ldots, p_{k+m} \in W - F$ and $p_{k+m+1} = f$, $z_i \in (V - T)^*$ for $i = 1, \ldots, k$, $y_j \in T^*$ for $j = 1, \ldots, m$, and $h = y_1y_2 \cdots y_{m-1}y_m$.

Informally, the queue grammar Q in Lemma 2 generates every string in $L(Q)$ so that it passes through state $\S$. Before it enters $\S$, it generates only strings over $V - T$; after entering $\S$, it generates only strings over T.

3 Definitions and Examples

In this section, we define controlled pure grammar systems and illustrate them by an example. Each definition is preceded by an intuitive explanation.

Informally, these systems are composed of n components, where $n \geq 1$, and a single alphabet. Every component contains (1) a set of rewriting rules over the alphabet, each having a single symbol on its left-hand side, and (2) a start string, from which these systems start their computation. Every rule is labelled by a unique label. Control languages for these systems are then defined over the set of all rule labels.

Definition 3. An *n-component pure grammar system* (an *n-pGS* for short), for some $n \geq 1$, is a $(2n + 2)$-tuple,

$$\varGamma = \big(T, \varPsi, P_1, w_1, P_2, w_2, \ldots, P_n, w_n\big)$$

where T and Ψ are two disjoint alphabets, $w_i \in T^*$, and $P_i \in \Psi \times T \times T^*$ for $i = 1, 2, \ldots, n$ are finite relations such that

(1) if $(r, a, x), (s, a, x) \in P_i$, then $r = s$;

(2) if $(r, a, x), (s, b, y) \in \bigcup_{1 \leq j \leq n} P_j$, where $a \neq b$ or $x \neq y$, then $r \neq s$;

The components Ψ, P_i, and w_i are called the alphabet of *rule labels*, the set of *rules* of the ith component, and the *start string* of the ith component, respectively. □

By analogy with context-free grammars, each rule (r, a, x) is written as $r\colon a \to x$ throughout this paper.

A configuration of Γ is an n-tuple of strings. It represents an instantaneous description of Γ. The initial configuration is formed by start strings.

Definition 4. Let $\Gamma = (T,\ \Psi,\ P_1,\ w_1,\ P_2,\ w_2,\ \ldots,\ P_n,\ w_n)$ be an n-pGS, for some $n \geq 1$. An n-tuple $(x_1,\ x_2,\ \ldots,\ x_n)$, where $x_i \in T^*$ for $i = 1, 2, \ldots, n$, is called a *configuration* of Γ. The configuration $(w_1,\ w_2,\ \ldots,\ w_n)$ is said to be *initial.* □

At every computational step, a rule from some component i is selected, and it is applied to the leftmost symbol of the ith string in the current configuration. Other strings remain unchanged. Hence, these systems work in a sequential way.

Definition 5. Let $\Gamma = (T,\ \Psi,\ P_1,\ w_1,\ P_2,\ w_2,\ \ldots,\ P_n,\ w_n)$ be an n-pGS, for some $n \geq 1$, and let $(x_1,\ x_2,\ \ldots,\ x_n)$, $(z_1,\ z_2,\ \ldots,\ z_n)$ be two configurations of Γ. The *direct derivation relation* over $(T^*)^n$, symbolically denoted by $\Rightarrow$, is defined as

$$(x_1, x_2, \ldots, x_n) \Rightarrow (z_1, z_2, \ldots, z_n)\ [r]$$

if and only if $r\colon a \to y \in P_i$, $x_i = av$, $z_i = yv$, where $v \in T^*$, for some $i \in \{1, 2, \ldots, n\}$, and $z_j = x_j$ for every $j \neq i$; $(x_1, x_2, \ldots, x_n) \Rightarrow (z_1, z_2, \ldots, z_n)\ [r]$ is simplified to $(x_1, x_2, \ldots, x_n) \Rightarrow (z_1, z_2, \ldots, z_n)$ if r is immaterial. In the standard manner, we extend $\Rightarrow$ to $\Rightarrow^m$, where $m \geq 0$; then, based on $\Rightarrow^m$, we define $\Rightarrow^*$. □

In the language generated by Γ, we include every string z satisfying the following two conditions: (1) it appears in the first component in a configuration that can be computed from the initial configuration, and (2) when it appears, all other strings are empty.

Definition 6. Let $\Gamma = (T,\ \Psi,\ P_1,\ w_1,\ P_2,\ w_2,\ \ldots,\ P_n,\ w_n)$ be an n-pGS, for some $n \geq 1$. The *language generated by* Γ is denoted by $L(\Gamma)$ and defined as

$$L(\Gamma) = \big\{z \in T^* \mid (w_1, w_2, \ldots, w_n) \Rightarrow^* (z, \varepsilon, \ldots, \varepsilon)\big\}$$ □

To control Γ, we define a language, Ξ, over its set of rule labels, and require that every successful computation—that is, a computation leading to a string in the generated language—is made by a sequence of rules in Ξ.

Definition 7. Let $\Gamma = (T, \Psi, P_1, w_1, P_2, w_2, \ldots, P_n, w_n)$ be an n-pGS, for some $n \geq 1$, and let $\Xi \subseteq \Psi^*$ be a *control language*. The *language generated by* Γ *with* Ξ is denoted by $L(\Gamma, \Xi)$ and defined as

$$L(\Gamma, \Xi) = \big\{ z \in T^* \mid (w_1, w_2, \ldots, w_n) \Rightarrow^* (z, \varepsilon, \ldots, \varepsilon)\ [\varrho] \text{ with } \varrho \in \Xi \big\}$$

If Ξ is regular, then the pair (Γ, Ξ) is called a *regular-controlled* n*-pGS*. □

Next, we illustrate the previous definitions by an example.

Example 1. Consider the 4-pGS

$$\Gamma = \big(\{a, b, c\}, \{r_i \mid 1 \leq i \leq 11\}, P_1, c, P_2, a, P_3, a, P_4, a\big)$$

where

$$\begin{aligned}
P_1 &= \{r_1\colon c \to cc,\ r_2\colon c \to bc,\ r_3\colon b \to bb,\ r_4\colon b \to ab,\ r_5\colon a \to aa\},\\
P_2 &= \{r_6\colon a \to aa,\ r_7\colon a \to \varepsilon\},\\
P_3 &= \{r_8\colon a \to aa,\ r_9\colon a \to \varepsilon\},\\
P_4 &= \{r_{10}\colon a \to aa,\ r_{11}\colon a \to \varepsilon\}.
\end{aligned}$$

Let $\Xi = \{r_6 r_8 r_{10}\}^* \{r_7 r_1\}^* \{r_2\} \{r_9 r_3\}^* \{r_4\} \{r_{11} r_5\}^*$ be a control language. Observe that every successful derivation in Γ with Ξ is of the form

$$\begin{array}{lll}
(c,\, a,\, a,\, a) \Rightarrow^{3(k-1)} & (c,\, a^k,\, a^k,\, a^k) & [(r_6 r_8 r_{10})^{k-1}]\\
\Rightarrow^{2k} & (c^{k+1},\, \varepsilon,\, a^k,\, a^k) & [(r_7 r_1)^k]\\
\Rightarrow & (bc^{k+1},\, \varepsilon,\, a^k,\, a^k) & [r_2]\\
\Rightarrow^{2k} & (b^{k+1}c^{k+1},\, \varepsilon,\, \varepsilon,\, a^k) & [(r_9 r_3)^k]\\
\Rightarrow & (ab^{k+1}c^{k+1},\, \varepsilon,\, \varepsilon,\, a^k) & [r_4]\\
\Rightarrow^{2k} & (a^{k+1}b^{k+1}c^{k+1},\, \varepsilon,\, \varepsilon,\, \varepsilon) & [(r_{11} r_5)^k]
\end{array}$$

for some $k \geq 1$. Clearly, $L(\Gamma, \Xi) = \{a^n b^n c^n \mid n \geq 2\}$. □

From Example 1, we see that regular-controlled pGSs can generate non-context-free languages. Moreover, notice that Ξ in Example 1 is, in fact, a union-free regular language (see [Nagy 2006]).

For every $n \geq 1$, let $\mathscr{L}(\mathrm{pGS}, n)$ denote the language family generated by n-pGSs. Define

$$\mathscr{L}(\mathrm{pGS}) = \bigcup_{n \geq 1} \mathscr{L}(\mathrm{pGS}, n)$$

4 Results

In this section, we prove results I through III, given next.

I. pGSs without control languages characterize only a proper subset of the family of context-free languages (Theorem 11).

II. Any recursively enumerable language can be generated by a regular-controlled 2-pGS (Theorems 13 and 14).

III. pGSs over unary alphabets controlled by languages from $\mathscr{L}(\mathrm{rC})$ generate only regular languages (Theorem 15).

4.1 Power of Pure Grammar Systems

First, we show that pGSs without control languages characterize only a proper subset of the family of context-free languages.

Lemma 8. *Let $\mathit{\Gamma}$ be an n-pGS satisfying $L(\mathit{\Gamma}) \neq \emptyset$, for some $n \geq 1$. Then, there is a 1-pGS $\mathit{\Omega}$ such that $L(\mathit{\Omega}) = L(\mathit{\Gamma})$.*

Proof. Let $\mathit{\Gamma} = (T, \mathit{\Psi}, P_1, w_1, P_2, w_2, \dots, P_n, w_n)$ be an n-pGS satisfying $L(\mathit{\Gamma}) \neq \emptyset$, for some $n \geq 1$. Let $z \in L(\mathit{\Gamma})$. By the definition of $L(\mathit{\Gamma})$, there exists

$$(w_1, w_2, \dots, w_n) \Rightarrow^* (z, \varepsilon, \dots, \varepsilon)$$

Since all components are independent from each other, there is also

$$(w_1, w_2, \dots, w_n) \Rightarrow^* (w_1, \varepsilon, \dots, \varepsilon) \Rightarrow^* (z, \varepsilon, \dots, \varepsilon)$$

Therefore, the 1-pGS $\mathit{\Omega} = (T, \mathit{\Psi}, P_1, w_1)$ clearly satisfies $L(\mathit{\Omega}) = L(\mathit{\Gamma})$. Hence, the lemma holds. □

Lemma 9. *Let $\mathit{\Gamma}$ be a 1-pGS. Then, $L(\mathit{\Gamma})$ is context-free.*

Proof. Let $\mathit{\Gamma} = (T, \mathit{\Psi}, P, w)$ be a 1-pGS. We next construct an extended push-down automaton M such that $L(M) = L(\mathit{\Gamma})$. Construct

$$M = \big(W, T, \mathit{\Omega}, R, s, \#, F\big)$$

as follows. Initially, set $W = \{s, t, f\}$, $\mathit{\Omega} = T \cup \{\#\}$, $R = \emptyset$, and $F = \{f\}$ (without any loss of generality, assume that $\# \notin T$). Perform (1) through (4), given next:

(1) add $\#s \rightarrow \mathrm{rev}(w)t$ to R;

(2) for each $a \rightarrow y \in P$, add $at \rightarrow \mathrm{rev}(y)t$ to R;

(3) add $t \to f$ to R;

(4) for each $a \in T$, add $afa \to f$ to R.

M works in the following way. It starts from $\#sz$, where $z \in T^*$. By the rule from (1), it generates the reversed version of the start string of Γ on the pushdown, ending up in $\mathrm{rev}(w)tz$. Then, by rules from (2), it rewrites $\mathrm{rev}(w)$ to a string over T. During both of these generations, no input symbols are read. To accept z, M has to end up in $\mathrm{rev}(z)tz$. After that, it moves to f by the rule from (3). Then, by using rules introduced in (4), it compares the contents of the pushdown with the input string. M accepts z if and only if the contents of the pushdown match the input string, meaning that $z \in L(\Gamma)$.

Clearly, $L(M) = L(\Gamma)$, so the lemma holds. □

Lemma 10. *There is no n-pGS that generates $\{a, aa\}$, for any $n \geq 1$.*

Proof. By contradiction. Without any loss of generality, by Lemma 8, we can only consider 1-pGSs. For the sake of contradiction, assume that there exists a 1-pGS, $\Omega = (\{a\}, P, w)$, such that $L(\Omega) = \{a, aa\}$. Observe that either (i) $w = a$ or (ii) $w = aa$. These two cases are discussed next.

(i) Assume that $w = a$. Then, there has to be $a \to aa \in P$. However, this implies that $L(\Omega)$ is infinite—a contradiction.

(ii) Assume that $w = aa$. Then, there has to be $a \to \varepsilon \in P$. However, this implies that $\varepsilon \in L(\Omega)$—a contradiction.

Hence, no 1-pGS that generates $\{a, aa\}$ exists, so the lemma holds. □

Theorem 11. $\mathscr{L}(\mathrm{pGS}) \subset \mathscr{L}(\mathrm{CF})$

Proof. Let Γ be an n-pGS, for some $n \geq 1$. If $L(\Gamma) = \emptyset$, then $L(\Gamma)$ is clearly context-free. Therefore, assume that $L(\Gamma) \neq \emptyset$. Then, by Lemma 8, there is a 1-pGS Ω such that $L(\Omega) = L(\Gamma)$. By Lemma 9, $L(\Omega)$ is context-free. Hence, $\mathscr{L}(\mathrm{pGS}) \subseteq \mathscr{L}(\mathrm{CF})$. By Lemma 10, $\mathscr{L}(\mathrm{CF}) - \mathscr{L}(\mathrm{pGS}) \neq \emptyset$, so the theorem holds. □

4.2 Power of Controlled Pure Grammar Systems

In this section, we prove that every recursively enumerable language can be generated by a regular-controlled 2-pGS.

Lemma 12. *Let Q be a left-extended queue grammar satisfying*

$$\mathrm{card}\big(\mathrm{alph}\big(L(Q)\big)\big) \geq 2$$

and the properties given in Lemma 2. Then, there is a 2-pGS Γ and a regular language Ξ such that $L(\Gamma, \Xi) = K$.

Proof. Let $Q = (W, V, T, R, s, F)$ be a left-extended queue grammar satisfying $\mathrm{card}(\mathrm{alph}(L(Q))) \geq 2$ and the properties given in Lemma 2. Let $s = a_0p_0$, $W = X \cup Y \cup \{\S\}$, and $F = \{f\}$. Assume that $\{0, 1\} \subseteq \mathrm{alph}(L(Q)))$. Observe that there exist a positive integer n and an injection ι from VW to $\{0, 1\}^n - 1^n$ so that ι remains an injection when its domain is extended to $(VW)^*$ in the standard way (after this extension, ι thus represents an injective homomorphism from $(VW)^*$ to $(\{0, 1\}^n - 1^n)^*$); a proof of this observation is simple and left to the reader. Based on ι, define the substitution ν from V to $(\{0, 1\}^n - 1^n)$ as $\nu(a) = \{\iota(aq) \mid q \in W\}$ for every $a \in V$. Extend the domain of ν to V^*. Furthermore, define the substitution μ from W to $(\{0, 1\}^n - 1^n)$ as $\mu(q) = \{\iota(aq) \mid a \in V\}$ for every $q \in W$. Extend the domain of μ to W^*.

Construct the 2-pGS

$$\Gamma = \bigl(T, \Psi, P_1, w_1, P_2, w_2\bigr)$$

where

$$\begin{aligned}
\Psi &= \{{}_{1y}^{1}1 \mid (a, p, y, q) \in R\}\\
&\quad \cup \{{}_{1w}^{1}1 \mid w \in \nu(y), (a, p, y, q) \in R\}\\
&\quad \cup \{{}_{1z}^{2}1 \mid z \in \mu(q), (a, p, y, q) \in R\}\\
&\quad \cup \{{}_{\varepsilon}^{i}0, {}_{\varepsilon}^{i}1 \mid i = 1, 2\}\\
&\quad \cup \{{}_{1^{n+1}}^{1}1\},\\
P_1 &= \{{}_{1y}^{1}1\colon 1 \to 1y \mid (a, p, y, q) \in R\}\\
&\quad \cup \{{}_{1w}^{1}1\colon 1 \to 1w \mid w \in \nu(y), (a, p, y, q) \in R\}\\
&\quad \cup \{{}_{\varepsilon}^{1}0\colon 0 \to \varepsilon, {}_{\varepsilon}^{1}1\colon 1 \to \varepsilon\}\\
&\quad \cup \{{}_{1^{n+1}}^{1}1\colon 1 \to 1^{n+1}\},\\
w_1 &= 1,\\
P_2 &= \{{}_{1z}^{2}1\colon 1 \to 1z \mid z \in \mu(q), (a, p, y, q) \in R\}\\
&\quad \cup \{{}_{\varepsilon}^{2}0\colon 0 \to \varepsilon, {}_{\varepsilon}^{2}1\colon 1 \to \varepsilon\},\\
w_2 &= 1^{n+1}.
\end{aligned}$$

Intuitively, ${}_{y}^{i}a$ means that a is rewritten to y in the ith component. Construct the right-linear grammar

$$G = \bigl(N, \Psi, P, \langle f, 2\rangle\bigr)$$

as follows. Initially, set $P = \emptyset$ and $N = \{\$\} \cup \{\langle p, i\rangle \mid p \in W, i = 1, 2\}$, where \$ is a new symbol. Perform (1) through (5), given next:

(1) if $(a, p, y, q) \in R$, where $a \in V - T$, $p \in W - F$, $q \in W$, and $y \in T^*$, add $\langle q, 2\rangle \to {}_{1y}^{1}1\ {}_{1z}^{1}1\ \langle p, 2\rangle$ to P for each $z \in \mu(p)$;

(2) add $\langle \S, 2\rangle \to {}_{1^{n+1}}^{1}1\ \langle \S, 1\rangle$ to P;

(3) if $(a, p, y, q) \in R$, where $a \in V - T$, $p \in W - F$, $q \in W$, and $y \in (V - T)^*$, add $\langle q, 1\rangle \to {}_{1w}^{1}1\ {}_{1z}^{1}1\ \langle p, 1\rangle$ to P for each $w \in \nu(y)$ and $z \in \mu(p)$;

(4) add $\langle p_0, 1\rangle \to {}^{1}_{1w}1\ \$$ to P for each $w \in \nu(a_0)$;

(5) add $\$ \to {}^{1}_{\varepsilon}0\ {}^{2}_{\varepsilon}0\ \$$, $\$ \to {}^{1}_{\varepsilon}1\ {}^{2}_{\varepsilon}1\ \$$, and $\$ \to \varepsilon$ to P.

Let $\mathcal{G} = (\Gamma, L(G))$. Before we establish the identity $L(\mathcal{G}) = L(Q)$, we explain how $\mathcal{G}$ works. In what follows, $(x, y)\ p$ denotes that the current configuration of Γ is (x, y) and that p is the nonterminal in the current sentential form of G. Recall the form of the derivations of Q in Lemma 2. The regular-controlled 2-pGS $\mathcal{G}$ simulates these derivations in reverse as follows. The start configuration of $\mathcal{G}$ is

$$\left(1, 1^{n+1}\right) \quad \langle f, 2\rangle$$

Rules from (1) generate $h \in L(Q)$ in the first component and encoded states p_{k+m}, p_{k+m-1}, $\dots$, p_k in the second component:

$$\left(1h, 1z1^n\right) \quad \langle \S, 2\rangle$$

where $z \in \mu(p_k p_{k+1} \cdots p_{k+m})$. Rules from (2) appends 1^n (a delimiter) to the first component:

$$\left(1^{n+1}h, 1z1^n\right) \quad \langle \S, 1\rangle$$

Rules from (3) generate encoded symbols a_{k+m}, a_{k+m-1}, $\dots$, a_1 in the first component and encoded states p_{k-1}, p_{k-2}, $\dots$, p_0 in the second component:

$$\left(1w1^nh, 1z'z1^n\right) \quad \langle p_0, 1\rangle$$

where $w \in \nu(a_1 a_2 \cdots a_{k+m})$ and $z' \in \mu(p_0 p_1 \cdots p_{k-1})$. A rule from (4) generates an encoded start symbol of Q, a_0:

$$\left(1w'w1^nh, 1z'z1^n\right) \quad \$$$

where $w' \in \nu(a_0)$. Notice that

$$w'w \in \nu(a_0 a_1 a_2 \cdots a_{k+m})$$

and

$$z'z \in \mu(p_0 p_1 p_2 \cdots p_{k+m})$$

Finally, rules from (5) check that $1w'w1^n = 1z'z1^n$ by erasing these two strings in a symbol-by-symbol way, resulting in (h, ε).

For brevity, the following proof omits some obvious details, which the reader can easily fill in. The next claim proves the above explanation rigorously—that is, it shows how $\mathcal{G}$ generates each string of $L(\mathcal{G})$.

Claim 1 *The regular-controlled 2-pGS $\mathcal{G}$ generates every $h \in L(\mathcal{G})$ in this way:*

$$
\begin{aligned}
& (1, 1^{n+1}) \\
\Rightarrow\ & (1y_m, 1g_{k+m}1^n) \\
\Rightarrow\ & (1y_{m-1}y_m, 1g_{k+m-1}g_{k+m}1^n) \\
& \quad \vdots \\
\Rightarrow\ & (1y_1 \cdots y_{m-1}y_m, 1g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow\ & (1^{n+1}h, 1g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow\ & (1t_{k+m}1^n h, 1g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow\ & (1t_{k+m-1}t_{k+m}1^n h, 1g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
& \quad \vdots \\
\Rightarrow\ & (1t_1 \cdots t_{k+m-1}t_{k+m}1^n h, 1g_0 \cdots g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow\ & (1t_0t_1 \cdots t_{k+m-1}t_{k+m}1^n h, 1g_0 \cdots g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow\ & (v_1h, v_1) \\
\Rightarrow\ & (v_2h, v_2) \\
& \quad \vdots \\
\Rightarrow\ & (v_\ell h, v_\ell) \\
\Rightarrow\ & (h, \varepsilon)
\end{aligned}
$$

where $k, m \geq 1$*;* $h = y_1 \cdots y_{m-1}y_m$*, where* $y_i \in T^*$ *for* $i = 1, 2, \ldots, m$*;* $t_i \in \nu(a_i)$ *for* $i = 0, 1, \ldots, k+m$*, where* $a_i \in V - T$*;* $g_i \in \mu(p_i)$ *for* $i = 0, 1, \ldots, k+m$*, where* $p_i \in W - F$*;* $v_i \in \{0, 1\}^*$ *for* $i = 1, 2, \ldots, \ell$*, where* $\ell = |t_0t_1 \cdots t_{k+m-1}t_{k+m}1^n|$*;* $|v_{i+1}| = |v_i| - 1$ *for* $i = 0, 1, \ldots, \ell - 1$*.*

Proof. Examine the construction of $\mathcal{G}$. Notice that in every successful computation, $\mathcal{G}$ uses rules from step (i) before it uses rules from step (i+1), for $i = 1, 2, 3, 4$. Thus, in a greater detail, every successful computation

$$(1, 1^{n+1}) \Rightarrow^* (h, \varepsilon)\ [\varrho] \text{ in } \mathcal{G}$$

where $\varrho \in L(G)$, can be expressed as

$$
\begin{aligned}
& (1, 1^{n+1}) \\
\Rightarrow\ & (1y_m, 1g_{k+m}1^n) \\
\Rightarrow\ & (1y_{m-1}y_m, 1g_{k+m-1}g_{k+m}1^n) \\
& \quad \vdots \\
\Rightarrow\ & (1y_1 \cdots y_{m-1}y_m, 1g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow\ & (1^{n+1}h, 1g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow\ & (1t_{k+m}1^n h, 1g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow\ & (1t_{k+m-1}t_{k+m}1^n h, 1g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
& \quad \vdots \\
\Rightarrow\ & (1t_1 \cdots t_{k+m-1}t_{k+m}1^n h, 1g_0 \cdots g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow\ & (1t_0t_1 \cdots t_{k+m-1}t_{k+m}1^n h, 1g_0 \cdots g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow^*\ & (h, \varepsilon)
\end{aligned}
$$

where $k, m \geq 1$; $h = y_1 \cdots y_{m-1} y_m$, where $y_i \in T^*$ for $i = 1, 2, \ldots, m$; $t_i \in \nu(a_i)$ for $i = 0, 1, \ldots, k + m$, where $a_i \in V - T$; $g_i \in \mu(p_i)$ for $i = 0, 1, \ldots, k + m$, where $p_i \in W - F$. Furthermore, during

$$(1t_0t_1 \cdots t_{k+m-1}t_{k+m}1^n h, 1g_0 \cdots g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \Rightarrow^* (h, \varepsilon)$$

only rules from (5) are used. Therefore,

$$1t_0t_1 \cdots t_{k+m-1}t_{k+m}1^n h \; = \; 1g_0 \cdots g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n$$

Let $v = 1t_0t_1 \cdots t_{k+m-1}t_{k+m}1^n h$. By rules from (5), $\mathcal{G}$ makes $|v|$ steps to erase v. Consequently, $(vh, v) \Rightarrow^* (h, \varepsilon)$ can be expressed as

$$\begin{aligned} & (vh, v) \\ \Rightarrow\ & (v_1 h, v_1) \\ \Rightarrow\ & (v_2 h, v_2) \\ & \quad \vdots \\ \Rightarrow\ & (v_\ell h, v_\ell) \\ \Rightarrow\ & (h, \varepsilon) \end{aligned}$$

where $v_i \in \{0, 1\}^*$ for $i = 1, 2, \ldots, \ell$, where $\ell = |v| - 1$, and $|v_{i+1}| = |v_i| - 1$ for $i = 0, 1, \ldots, \ell - 1$. As a result, the claim holds. □

Let $\mathcal{G}$ generate $h \in L(\mathcal{G})$ in the way described in Claim 1. Examine the construction of G to see that at this point, R contains $(a_0, p_0, z_0, p_1), \ldots, (a_k, p_k, z_k, p_{k+1}), (a_{k+1}, p_{k+1}, y_1, p_{k+2}), \ldots, (a_{k+m-1}, p_{k+m-1}, y_{m-1}, p_{k+m}), (a_{k+m}, p_{k+m}, y_m, p_{k+m+1})$, where $p_{k+m+1} = f$ and $z_i \in (V - T)^*$ for $i = 1, 2, \ldots, k$, so Q makes the generation of h in the way described in Lemma 2. Thus, $h \in L(Q)$. Consequently, $L(\mathcal{G}) \subseteq L(Q)$.

Let Q generate $g \in L(Q)$ in the way described in Lemma 2. Then, $\mathcal{G}$ generates h in the way described in Claim 1, so $L(Q) \subseteq L(\mathcal{G})$; a detailed proof of this inclusion is left to the reader.

As $L(\mathcal{G}) \subseteq L(Q)$ and $L(Q) \subseteq L(\mathcal{G})$, $L(\mathcal{G}) = L(Q)$. Hence, the lemma holds. □

Theorem 13. *Let K be a recursively enumerable language satisfying*

$$\mathrm{card}\big(\mathrm{alph}(K)\big) \geq 2$$

Then, there is a 2-pGS Γ and a regular language Ξ such that $L(\Gamma, \Xi) = K$.

Proof. This theorem follows from Lemmas 2 and 12. □

Theorem 14. *Let K be a unary recursively enumerable language, and let $c \notin \mathrm{alph}(K)$ be a new symbol. Then, there is a 2-pGS Γ and a regular language Ξ such that $L(\Gamma, \Xi) = K$.*

Proof. This theorem can be proved by analogy with the proof of Theorem 13 (we use c as the second symbol in the proof of Lemma 12). □

4.3 Power of Controlled Pure Grammar Systems Over Unary Alphabets

In this section, we prove that pGSs over unary alphabets controlled by languages from $\mathscr{L}(\mathrm{rC})$ generate only regular languages.

Theorem 15. *Let $\mathit{\Gamma} = (T,\ \mathit{\Psi},\ P_1,\ w_1,\ P_2,\ w_2,\ \ldots,\ P_n,\ w_n)$ be an n-pGS satisfying* $\mathrm{card}(T) = 1$*, for some $n \geq 1$, and let $\mathit{\Xi} \in \mathscr{L}(\mathrm{rC})$. Then, $L(\mathit{\Gamma},\ \mathit{\Xi})$ is regular.*

Proof. Let $\mathit{\Gamma} = (T,\ \mathit{\Psi},\ P_1,\ w_1,\ P_2,\ w_2,\ \ldots,\ P_n,\ w_n)$ be an n-pGS satisfying $\mathrm{card}(T) = 1$, for some $n \geq 1$, and let $\mathit{\Xi} \in \mathscr{L}(\mathrm{rC})$. We show how to convert $\mathit{\Gamma}$ and $\mathit{\Xi}$ into an equivalent regular-controlled grammar $(G,\ \mathit{\Pi})$. Then, since $\mathrm{card}(T) = 1$, Lemma 1 implies that $L(\mathit{\Gamma},\ \mathit{\Xi})$ is regular.

Let $\bar{G} = (\bar{N},\ \mathit{\Psi},\ \bar{\mathit{\Phi}},\ \bar{P},\ \bar{S})$ be a context-free grammar and $\bar{M}$ be a finite automaton such that $L(\bar{G}, L(\bar{M})) = \mathit{\Xi}$. Let $T = \{c\}$. To distinguish between the components of $\mathit{\Gamma}$ in G, we encode c for each component. Set

$$N_{\#} = \{c_i \mid 1 \leq i \leq n\}$$

For each $i \in \{1, \ldots, n\}$, define the homomorphism τ_i from T^* to $N_{\#}^*$ as $\tau_i(c) = c_i$. For each $i \in \{1, \ldots, n\}$, set

$$R_i = \{r\colon \tau_i(c) \to \tau_i(y) \mid r\colon c \to y \in P_i\}$$

Define G as

$$G = \big(N, \{c\}, \mathit{\Phi}, R, S\big)$$

where

$$\begin{aligned}
N &= \{S\} \cup \bar{N} \cup \mathit{\Psi} \cup N_{\#} \cup \textstyle\bigcup_{1 \leq i \leq n} R_i,\\
\mathit{\Phi} &= \bar{\mathit{\Phi}} \cup \{s, c_1\} \cup \{r_\varepsilon \mid r \in \bar{\mathit{\Phi}}\},\\
R &= \bar{P} \cup \{s\colon S \to \bar{S}\tau_1(w_1)\tau_2(w_2)\cdots\tau_n(w_n)\}\\
&\quad \cup \{c_1\colon c_1 \to c\}\\
&\quad \cup \{r_\varepsilon\colon r \to \varepsilon \mid r \in \bar{\mathit{\Phi}}\},\\
\mathit{\Lambda} &= \{r_\varepsilon r \mid r\colon a \to y \in \textstyle\bigcup_{1 \leq i \leq n} R_i\}^*,\\
\mathit{\Pi} &= \{s\}L(\bar{M})\mathit{\Lambda}\{c_1\}^* \text{ (the control language of } G).
\end{aligned}$$

Without any loss of generality, we assume that $\{S\}$, $\bar{N}$, $\mathit{\Psi}$, $N_{\#}$, and $\bigcup_{1 \leq i \leq n} R_i$ are pairwise disjoint, and we also assume that $\bar{\mathit{\Phi}}$, $\{s, c_1\}$, and $\{r_\varepsilon \mid r \in \bar{\mathit{\Phi}}\}$ are pairwise disjoint.

In every successful derivation of every $z \in L(G,\ \mathit{\Pi})$ in G, s is applied to S. It generates the start symbol of $\bar{G}$ and encoded start strings of each component of $\mathit{\Gamma}$. Indeed, instead of c, we generate c_i, where $1 \leq i \leq n$. Then, rules from $\bar{P}$ are used to rewrite $\bar{S}$ to a control string from $L(\bar{G},\ L(\bar{M}))$. Using pairs of rules r_ε $r \in \Lambda$, G erases an occurrence of r in the current sentential form, and applies r to a symbol in a proper substring of the current sentential corresponding to the component which would use r. This process is repeated until the control string is completely erased. Since $\mathrm{card}(T) = 1$, the order of used rules and the occurrence of the rewritten c are not important. Finally, G uses $c_1 \colon c_1 \to c$ to decode each occurrence of c_1 back to c, thus obtaining z. If G applies its rules in an improper way—that is, if there remain some symbols from $\bigcup_{1 \le i \le n} R_i$ or from $\{c_i \mid 2 \le i \le n\}$ after the last pair from Λ is applied—the derivation is blocked.

Based on these observations, we see that every successful derivation of every $z \in L(G, \mathit{\Pi})$ in G with $\mathit{\Pi}$ is of the form

$$\begin{aligned} S \Rightarrow\ & \bar{S}\tau_1(w_1)\tau_2(w_2)\cdots\tau_n(w_n) && [s] \\ \Rightarrow^*\ & \varrho\tau_1(w_1)\tau_2(w_2)\cdots\tau_n(w_n) && [\upsilon] \\ \Rightarrow^*\ & \tau_1(z) && [\lambda] \\ \Rightarrow^*\ & z && [\gamma] \end{aligned}$$

where $\varrho \in L(\bar{G}, L(\bar{M}))$, $\upsilon \in L(\bar{M})$, $\lambda \in \mathit{\Lambda}$, and $\gamma \in \{c_1\}^*$. In $\mathit{\Gamma}$, there is

$$(w_1, w_2, \dots, w_n) \Rightarrow^* (z, \varepsilon, \dots, \varepsilon)\ [\varrho]$$

Hence, $L(G, \mathit{\Pi}) \subseteq L(\mathit{\Gamma})$. Conversely, for every $z \in L(\mathit{\Gamma})$, there is a derivation of z in G with $\mathit{\Pi}$ of the above form, so $L(\mathit{\Gamma}) \subseteq L(G, \mathit{\Pi})$. Therefore, $L(\mathit{\Gamma}) = L(G, \mathit{\Pi})$, and the theorem holds. A rigorous proof of the identity $L(\mathit{\Gamma}) = L(G, \mathit{\Pi})$ is left to the reader. □

From Theorem 15, we obtain a corollary concerning regular-controlled n-pGSs over unary alphabets, stated next.

Corollary 16. *Let $\mathit{\Gamma} = (T, \mathit{\Psi}, P_1, w_1, P_2, w_2, \dots, P_n, w_n)$ be an n-pGS satisfying* $\mathrm{card}(T) = 1$*, for some $n \ge 1$, and let $\mathit{\Xi} \in \mathit{\Psi}^*$ be regular. Then, $L(\mathit{\Gamma}, \mathit{\Xi})$ is regular.* □

Notice that this result is surprising in the light of Theorem 13, which says that every recursively enumerable language over an alphabet with at least two symbols can be generated by a regular-controlled 2-pGS.

5 Concluding Remarks

The next four open problem areas are related to the achieved results.

I. Let $\mathit{\Gamma}$ be an n-pGS, for some $n \ge 1$. By Lemma 9, $L(\mathit{\Gamma})$ is context-free. Is $L(\mathit{\Gamma})$, in fact, regular?

II. Consider proper subfamilies of the family of regular languages (see, for example, [Rozenberg and Salomaa 1997, Nagy 2006, Dassow and Truthe 2008, Bordihn et al. 2009]). Can we obtain Theorems 13 and 14 when the control languages are from these subfamilies?

III. By Theorems 13 and 14, two components suffice to generate any recursively enumerable language by regular-controlled pGSs. What is the power of controlled pGSs with a single component?

IV. Let Γ be an n-pGS, for some $n \geq 1$. If no rule of Γ has ε on its right-hand side, then Γ is said to be *propagating*. What is the power of controlled propagating pGSs?

Acknowledgments

This work was supported by the following grants: FRVŠ MŠMT FR271/2012/G1, BUT FIT-S-11-2, MŠMT CZ 1.05/1.1.00/02.0070, CEZ MŠMT MSM0021630528.

References

[Aho et al. 2006] Aho, A. V., Lam, M. S., Sethi, R., and Ullman, J. D.: "Compilers: Principles, Techniques, and Tools"; Addison-Wesley, Boston, 2nd edition (2006).

[Aydin and Bordihn 2003] Aydin, S. and Bordihn, H.: "Sequential versus parallel grammar formalisms with respect to measures of descriptional complexity"; Fundamenta Informaticae, 55, 3-4 (2003), 243–254.

[Beek and Kleijn 2002] Beek, M. and Kleijn, J.: "Petri net control for grammar systems"; In Formal and natural computing, New York, NY. Springer-Verlag (2002), 220–243.

[Bensch and Bordihn 2007] Bensch, S. and Bordihn, H.: "Active symbols in pure systems"; Fundamenta Informaticae, 76, 3, (2007), 239–254.

[Bordihn et al. 1999] Bordihn, H., Csuhaj-Varjú, E., and Dassow, J.: "CD grammar systems versus L systems"; In Grammatical Models of Multi-Agent Systems, volume 8 of *Topics in Computer Mathematics*, Amsterdam, NL. Gordon and Breach Science Publishers (1999), 18–32.

[Bordihn et al. 2009] Bordihn, H., Holzer, M., and Kutrib, M.: "Determination of finite automata accepting subregular languages"; Theoretical Computer Science, 410, 35, (2009), 3209–3222.

[Csuhaj-Varjú et al. 1994] Csuhaj-Varjú, E., Dassow, J., Kelemen, J., and Păun, G.: "Grammar Systems: A Grammatical Approach to Distribution and Cooperation"; Gordon and Breach, Yverdon (1994).

[Csuhaj-Varjú et al. 1993] Csuhaj-Varjú, E., Dassow, J., and Păun, G.: "Dynamically controlled cooperating/distributed grammar systems"; Information Sciences, 69, 1-2, (1993), 1–25.

[Csuhaj-Varjú and Vaszil 2001] Csuhaj-Varjú, E. and Vaszil, G.: "On context-free parallel communicating grammar systems: synchronization, communication, and normal forms"; Theoretical Computer Science, 255, 1-2, (2001), 511–538.

[Dassow and Păun 1989] Dassow, J. and Păun, G.: "Regulated Rewriting in Formal Language Theory"; Springer, New York (1989).

[Dassow and Truthe 2008] Dassow, J. and Truthe, B.: "Subregularly tree controlled grammars and languages"; In Automata and Formal Languages, Balatonfured, HU. Computer and Automation Research Institute, Hungarian Academy of Sciences (2008), 158–169.

[Fernau and Holzer 2002] Fernau, H. and Holzer, M.: "Graph-controlled cooperating distributed grammar systems with singleton components"; Journal of Automata, Languages and Combinatorics, 7, 4, (2002), 487–503.

[Goldefus 2009] Goldefus, F.: "Cooperating distributed grammar systems and graph controlled grammar systems with infinite number of components"; In Proceedings of the 15th Conference STUDENT EEICT 2009 Volume 4, Brno, CZ. Faculty of Information Technology BUT (2009), 400–404.

[Kleijn and Rozenberg 1983] Kleijn, H. C. M. and Rozenberg, G.: "On the generative power of regular pattern grammars"; Acta Informatica, 20 (1983), 391–411.

[Kolář and Meduna 2000] Kolář, D. and Meduna, A.: "Regulated pushdown automata"; Acta Cybernetica, 2000, 4 (2000), 653–664.

[Lukáš and Meduna 2006] Lukáš, R. and Meduna, A.: "Multigenerative grammar systems"; Schedae Informaticae, 2006, 15 (2006), 175–188.

[Lukáš and Meduna 2010] Lukáš, R. and Meduna, A.: "Multigenerative grammar systems and matrix grammars"; Kybernetika, 46, 1 (2010), 68–82.

[Mäkinen 1986] Mäkinen, E.: "A note on pure grammars"; Information Processing Letters, 23, 5 (1986), 271–274.

[Martín-Vide et al. 2004] Martín-Vide, C., Mitrana, V., and Păun, G., editors: "Formal Languages and Applications", chapter 13, pages 249–274; Springer, Berlin (2004).

[Martinek 1998] Martinek, P.: "Limits of pure grammars with monotone productions"; Fundamenta Informaticae, 33, 3 (1998), 265–280.

[Maurer et al. 1980] Maurer, H. A., Salomaa, A., and Wood, D.: "Pure grammars"; Information and Control, 44, 1 (1980), 47–72.

[Meduna 2000] Meduna, A.: "Automata and Languages: Theory and Applications"; Springer, London (2000).

[Meduna 2003] Meduna, A.: "Simultaneously one-turn two-pushdown automata"; International Journal of Computer Mathematics, 2003, 80 (2003), 679–687.

[Meduna 2004] Meduna, A.: "Two-way metalinear PC grammar systems and their descriptional complexity"; Acta Cybernetica, 2004, 16 (2004), 385–397.

[Meduna 2007] Meduna, A.: "Elements of Compiler Design"; Auerbach Publications, Boston (2007).

[Nagy 2006] Nagy, B.: "Union-free regular languages and 1-cycle-free-path automata"; Publicationes Mathematicae Debrecen, 98 (2006), 183–197.

[Păun 1993] Păun, G.: "On the synchronization in parallel communicating grammar systems"; Acta Informatica, 30, 4 (1993), 351–367.

[Rozenberg and Salomaa 1980] Rozenberg, G. and Salomaa, A.: "Mathematical Theory of L Systems"; Academic Press, Orlando (1980).

[Rozenberg and Salomaa 1986] Rozenberg, G. and Salomaa, A.: "The Book of L"; Springer-Verlag, New York (1986).

[Rozenberg and Salomaa 1997] Rozenberg, G. and Salomaa, A., editors: "Handbook of Formal Languages, Vol. 1: Word, Language, Grammar"; Springer, New York (1997).

[Salomaa 1973] Salomaa, A.: "Formal Languages"; Academic Press, London (1973).

[von zur Gathen and Gerhard 2003] von zur Gathen, J. and Gerhard, J.: "Modern Computer Algebra"; Cambridge University Press, New York, 2nd edition (2003).