Chapter 14
Controlled Pure Grammar Systems

Abstract This chapter introduces pure grammar systems, which have only terminals. They generate their languages in the leftmost way, and in addition, this generative process is regulated by control languages over rule labels. The chapter concentrates its attention on investigating the generative power of these systems. It establishes three major results. First, without any control languages, these systems do not even generate some context-free languages. 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. Finally, this chapter 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 define nothing but regular languages. The chapter consists of two sections. First, Section 14.1 define controlled pure grammar systems and illustrate them by an example. Then, Section 14.2 rigorously establishes the results mentioned above and points out several open problems.

Key words: grammar systems, control languages, pure versions, leftmost derivations, generative power, language families

To grasp the discussion of the present chapter fully, we should realize that context-free grammars are quite central to formal language theory as a whole (see [10, 17, 23, 24]). 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 [7]), regulated context-free grammars (see Chapters 4 and 5), pure context-free grammars (see [14–16] and page 242 in [23]), and context-free grammars with leftmost derivations (see [19] and Section 5.1 in [17]) 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 the 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 grammars (see Section 3.3).

(IV) Context-free grammars that perform only leftmost derivations fulfill a key role in a principal application area of these grammars—parsing (see [1, 19]).

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 [2, 4, 5]). Similarly, based upon various combinations of (I) and (II), a number of regulated grammar systems were defined and discussed (see Chapter 13 and [3, 8, 9, 12, 13, 22]). Following this vivid investigation trend, the present chapter combines all the four modifications mentioned above.

More specifically, this chapter 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 chapter 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 14.2.4). 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 14.2.7 and 14.2.8). Finally, this chapter 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 define nothing but regular languages (Theorem 14.2.9).

The present chapter is organized as follows. First, Section 14.1 define controlled pure grammar systems and illustrate them by an example. Then, Section 14.2 rigorously establishes the results mentioned above. A formulation of several open problems closes the chapter.

14.1 Definitions and Examples

In this section, we define controlled pure grammar systems and illustrate them by an example.

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

Definition 14.1.1. An *n-component pure grammar system* (an *n-pGS* for short), for some $n \geq 1$, is a $(2n+2)$-tuple

$$\Gamma = \big(T, \Psi, P_1, w_1, P_2, w_2, \ldots, P_n, w_n\big)$$

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 chapter.

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

Definition 14.1.2. 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 the *start configuration*. □

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 14.1.3. 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 $\Rightarrow_\Gamma$, is defined as

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

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 $j \neq i$; $(x_1, x_2, \ldots, x_n) \Rightarrow_\Gamma (z_1, z_2, \ldots, z_n)\ [r]$ is simplified to $(x_1, x_2, \ldots, x_n) \Rightarrow_\Gamma (z_1, z_2, \ldots, z_n)$ if r is immaterial.

Let $\chi_0, \chi_1, \chi_2, \ldots, \chi_m$ be $m+1$ configurations such that

$$\chi_0 \Rightarrow_\Gamma \chi_1\ [r_1] \Rightarrow_\Gamma \chi_2\ [r_2] \Rightarrow_\Gamma \cdots \Rightarrow_\Gamma \chi_m\ [r_m]$$

by applying rules labeled with r_1 through r_m, for some $m \geq 1$. Then, we write

$$\chi_0 \Rightarrow_\Gamma^m \chi_m\ [r_1 \cdots r_m]$$

Moreover, for every configuration χ, we write $\chi \Rightarrow_\Gamma^0 \chi\ [\varepsilon]$. For any two configurations χ and χ', if $\chi \Rightarrow_\Gamma^m \chi'\ [\rho]$ for $m \geq 0$ and $\rho \in \Psi^*$, then we write

$$\chi \Rightarrow_\Gamma^* \chi'\ [\rho]$$

□

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 start configuration, and (2) when it appears, all the other strings are empty.

Definition 14.1.4. Let $\Gamma = (T, \Psi, P_1, w_1, P_2, w_2, \dots, P_n, w_n)$ be an n-pGS, for some $n \geq 1$. The *language* of Γ is denoted by $L(\Gamma)$ and defined as

$$L(\Gamma) = \big\{ z \in T^* \mid (w_1, w_2, \dots, w_n) \Rightarrow_{\Gamma}^{*} (z, \varepsilon, \dots, \varepsilon) \big\} \qquad \Box$$

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

Definition 14.1.5. Let $\Gamma = (T, \Psi, P_1, w_1, P_2, w_2, \dots, P_n, w_n)$ 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

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

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

Next, we illustrate the previous definitions by an example.

Example 14.1.6. 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

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

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

From Example 14.1.6, we see that regular-controlled pGSs can generate non-context-free languages. Moreover, notice that Ξ in Example 14.1.6 is, in fact, a union-free regular language (see [21]).

For every $n \geq 1$, let ${}_n\mathbf{pGS}$ denote the language family generated by n-pGSs. Furthermore, set

$$\mathbf{pGS} = \bigcup_{n \geq 1} {}_n\mathbf{pGS}$$

14.2 Generative Power

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 14.2.4).

(II) Every recursively enumerable language can be generated by a regular-controlled 2-pGS (Theorems 14.2.7 and 14.2.8).

(III) pGSs over unary alphabets controlled by languages from **rC** generate only the family of regular languages (Theorem 14.2.9).

Power of Pure Grammar Systems

First, we show that the language family generated by pGSs without control languages is properly included in the family of context-free languages.

Lemma 14.2.1. *Let Γ be an n-pGS satisfying $L(\Gamma) \neq \emptyset$, for some $n \geq 1$. Then, there is a 1-pGS Ω such that $L(\Omega) = L(\Gamma)$.*

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

$$(w_1, w_2, \ldots, w_n) \Rightarrow_{\Gamma}^{*} (z, \varepsilon, \ldots, \varepsilon)$$

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

$$(w_1, w_2, \ldots, w_n) \Rightarrow_{\Gamma}^{*} (w_1, \varepsilon, \ldots, \varepsilon) \Rightarrow_{\Gamma}^{*} (z, \varepsilon, \ldots, \varepsilon)$$

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

Lemma 14.2.2. *Let Γ be a 1-pGS. Then, $L(\Gamma)$ is context-free.*

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

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

as follows. Initially, set $W = \{s, t, f\}$, $\Omega = T \cup \{\#\}$, $R = \emptyset$, 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 $t \rightarrow f$ to R.
(4) For each $a \in T$, add $afa \rightarrow f$ 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, $L_{ef}(M) = L(\Gamma)$, so the lemma holds. □

Lemma 14.2.3. *There is no n-pGS that generates $\{a, aa\}$, for any $n \geq 1$.*

Proof. By contradiction. Without any loss of generality, making use of Lemma 14.2.1, 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, $a \rightarrow aa \in P$. However, this implies that $L(\Omega)$ is infinite—a contradiction.
(ii) Assume that $w = aa$. Then, $a \rightarrow \varepsilon \in P$. However, this implies that $\varepsilon \in L(\Omega)$—a contradiction.

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

Theorem 14.2.4. pGS $\subset$ 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 14.2.1, there is a 1-pGS Ω such that $L(\Omega) = L(\Gamma)$. By Lemma 14.2.2, $L(\Omega)$ is context-free. Hence, **pGS** $\subseteq$ **CF**. By Lemma 14.2.3, **CF** $-$ **pGS** $\neq \emptyset$, so the theorem holds. □

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.

To do this, we need the following normal form of left-extended queue grammars (see Definition 3.3.18).

Lemma 14.2.5 (see [18]). *For every recursively enumerable language K, there exists a left-extended queue grammar, $Q = (V, T, W, F, R, g)$, 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

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

where $k,m \geq 1$, $a_i \in V-T$ for $i = 0, \ldots, k+m$, $x_j \in (V-T)^*$ for $j = 1, \ldots, k+m$, $g = a_0p_0$, $a_jx_j = x_{j-1}z_j$ 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 14.2.5 generates every string in $L(Q)$ so that it passes through state §. Before it enters §, it generates only strings over $V-T$; after entering §, it generates only strings over T.

Lemma 14.2.6. *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 14.2.5. Then, there is a 2-pGS Γ and a regular language Ξ such that $L(\Gamma, \Xi) = K$.

Proof. Let $Q = (V, T, W, F, R, g)$ 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 14.2.5. Let $g = a_0p_0$, $W = X \cup Y \cup \{\S\}$, and $F = \{f\}$. Assume that

$$\{0,1\} \subseteq \mathrm{alph}\big(L(Q)\big)$$

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

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

where

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

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

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

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 $\langle q,2\rangle \rightarrow {}^{1}_{1y}1\ {}^{1}_{1z}1\ \langle p,2\rangle$ to P for each $z \in \mu(p)$.
(2) Add $\langle \S,2\rangle \rightarrow {}^{1}_{1^{n+1}}1\ \langle \S,1\rangle$ 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 $\langle q,1\rangle \rightarrow {}^{1}_{1w}1\ {}^{1}_{1z}1\ \langle p,1\rangle$ to P for each $w \in \nu(y)$ and $z \in \mu(p)$.
(4) Add $\langle p_0,1\rangle \rightarrow {}^{1}_{1w}1\ \$$ to P for each $w \in \nu(a_0)$.
(5) Add $\$ \rightarrow {}^{1}_{\varepsilon}0\ {}^{2}_{\varepsilon}0\ \$$, $\$ \rightarrow {}^{1}_{\varepsilon}1\ {}^{2}_{\varepsilon}1\ \$$, and $\$ \rightarrow \varepsilon$ to P.

Let $\mathscr{G} = (\Gamma, L(G))$. Before we establish the identity $L(\mathscr{G}) = L(Q)$, we explain how $\mathscr{G}$ works. In what follows, $(x,y)\ p$ means that the current configuration of Γ is (x,y) and that p is the nonterminal in the current sentential form of G. Consider the form of the derivations of Q in Lemma 14.2.5. The regular-controlled 2-pGS $\mathscr{G}$ simulates these derivations in reverse as follows. The start configuration of $\mathscr{G}$ is

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

Rules from (1) generate $h \in L(Q)$ in the first component and encoded states p_{k+m}, p_{k+m-1}, …, 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}, …, a_1 in the first component and encoded states p_{k-1}, p_{k-2}, …, p_0 in the second component

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

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

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

where $w' \in \nu(a_0)$ and a_0 is the start symbol of Q. 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) checks 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 $\mathscr{G}$ generates each string of $L(\mathscr{G})$.

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

$$\begin{array}{ll}
 & (1, 1^{n+1}) \\
\Rightarrow_\Gamma & (1y_m, 1g_{k+m}1^n) \\
\Rightarrow_\Gamma & (1y_{m-1}y_m, 1g_{k+m-1}g_{k+m}1^n) \\
 & \vdots \\
\Rightarrow_\Gamma & (1y_1 \cdots y_{m-1}y_m, 1g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow_\Gamma & (1^{n+1}h, 1g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow_\Gamma & (1t_{k+m}1^n h, 1g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow_\Gamma & (1t_{k+m-1}t_{k+m}1^n h, 1g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
 & \vdots \\
\Rightarrow_\Gamma & (1t_1 \cdots t_{k+m-1}t_{k+m}1^n h, 1g_0 \cdots g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow_\Gamma & (1t_0t_1 \cdots t_{k+m-1}t_{k+m}1^n h, 1g_0 \cdots g_{k-2}g_{k-1}g_k \cdots g_{k+m-1}g_{k+m}1^n) \\
\Rightarrow_\Gamma & (v_1 h, v_1)
\end{array}$$

$$\begin{aligned} &\Rightarrow_\Gamma\ (v_2h, v_2)\\ &\ \ \vdots\\ &\Rightarrow_\Gamma\ (v_\ell h, v_\ell)\\ &\Rightarrow_\Gamma\ (h, \varepsilon)\end{aligned}$$

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 $\mathscr{G}$. Notice that in every successful computation, $\mathscr{G}$ 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

$$(1, 1^{n+1}) \Rightarrow_\Gamma^* (h, \varepsilon)\ [\rho]$$

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

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

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

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

only rules from (5) are used. Therefore,

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

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

$$\begin{aligned}
& (vh, v) \\
\Rightarrow_\Gamma\ & (v_1h, v_1) \\
\Rightarrow_\Gamma\ & (v_2h, v_2) \\
& \vdots \\
\Rightarrow_\Gamma\ & (v_\ell h, v_\ell) \\
\Rightarrow_\Gamma\ & (h, \varepsilon)
\end{aligned}$$

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 $\mathscr{G}$ generate $h \in L(\mathscr{G})$ 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), …, (a_k, p_k, z_k, p_{k+1}), $(a_{k+1}, p_{k+1}, y_1, p_{k+2})$, …, $(a_{k+m-1}, p_{k+m-1}, y_{m-1}, p_{k+m})$, $(a_{k+m}, p_{k+m}, y_m, p_{k+m+1})$, 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 14.2.5. Thus, $h \in L(Q)$. Consequently, $L(\mathscr{G}) \subseteq L(Q)$.

Let Q generate $g \in L(Q)$ in the way described in Lemma 14.2.5. Then, $\mathscr{G}$ generates h in the way described in Claim 1, so $L(Q) \subseteq L(\mathscr{G})$; a detailed proof of this inclusion is left to the reader.

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

Theorem 14.2.7. *Let K be a recursively enumerable language satisfying*

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

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

Proof. This theorem follows from Lemmas 14.2.5 and 14.2.6. □

Theorem 14.2.8. *Let K be a unary recursively enumerable language, and let $c \notin \operatorname{alph}(K)$ 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 14.2.7 (we use c as the second symbol in the proof of Lemma 14.2.6). □

Power of Controlled Pure Grammar Systems Over Unary Alphabets

In this section, we prove that pGSs over unary alphabets controlled by languages from **rC** generate only regular languages.

Theorem 14.2.9. *Let $\Gamma = (T, \Psi, P_1, w_1, P_2, w_2, \ldots, P_n, w_n)$ be an n-pGS satisfying* $\operatorname{card}(T) = 1$, *for some $n \geq 1$, and let $\Xi \in$* **rC**. *Then, $L(\Gamma, \Xi)$ is regular.*

Proof. Let $\Gamma = (T, \Psi, P_1, w_1, P_2, w_2, \dots, P_n, w_n)$ be an n-pGS satisfying $\mathrm{card}(T) = 1$, for some $n \geq 1$, and let $\Xi \in \mathbf{rC}$. We show how to convert Γ and Ξ into an equivalent regular-controlled grammar (G, Π). Then, since $\mathrm{card}(T) = 1$, Theorem 5.1.6 implies that $L(\Gamma, \Xi)$ is regular.

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

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

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

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

Define G as

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

where

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

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

In every successful derivation of every $z \in L(G, \Pi)$ in G, s is applied to S. It generates the start symbol of $\bar{G}$ and encoded start strings of each component of Γ. 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 $L(\bar{G}, L(\bar{M}))$. By using pairs of rules $r_\varepsilon r \in \Lambda$, 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 cs are not important. Finally, G uses $c_1\colon c_1 \rightarrow c$ 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 $\bigcup_{1 \leq i \leq n} R_i$ or from $\{c_i \mid 2 \leq i \leq n\}$ after the last pair from Λ is applied—the derivation is blocked.

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

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

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

$$(w_1, w_2, \ldots, w_n) \Rightarrow_\Gamma^* (z, \varepsilon, \ldots, \varepsilon)\ [\rho]$$

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

From Theorem 14.2.9, we obtain the following corollary.

Corollary 14.2.10. *Let* $\Gamma = (T, \Psi, P_1, w_1, P_2, w_2, \ldots, P_n, w_n)$ *be an n-pGS satisfying* card$(T) = 1$*, for some* $n \geq 1$*, and let* $\Xi \in \Psi^*$ *be regular. Then,* $L(\Gamma, \Xi)$ *is regular.* □

Notice that this result is surprising in the light of Theorems 14.2.7 and 14.2.8, which say that 2-pGSs having at least two symbols are computationally complete.

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

Open Problem 14.2.11. Let Γ be an n-pGS, for some $n \geq 1$. By Lemma 14.2.2, $L(\Gamma)$ is context-free. Is $L(\Gamma)$, in fact, regular? □

Open Problem 14.2.12. Consider proper subfamilies of the family of regular languages (see [6, 11, 21, 23]). Can we obtain Theorems 14.2.7 and 14.2.8 when the control languages are from these subfamilies? □

Open Problem 14.2.13. By Theorems 14.2.7 and 14.2.8, two components suffice to generate any recursively enumerable language by regular-controlled pGSs. What is the power of controlled pGSs with a single component? □

Open Problem 14.2.14. 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? □

For some preliminary solutions to the above four open problems, see [20].

References

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

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

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

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

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

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

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

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

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

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

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

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

[13] 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, vol. 4, pp. 400–404. Brno University of Technology, Brno, CZ (2009)

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

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

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

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

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