P. Dömösi, Sz. Iván (Eds.): Automata and Formal Languages. AFL 2011,
Debrecen, Hungary, August 17–22, 2011. Proceedings, pages 316 – 328

On Nondeterminism in Programmed Grammars

Alexander Meduna Lukáš Vrábel Petr Zemek

Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
{meduna,ivrabel,izemek}@fit.vutbr.cz

Abstract

In the present paper, we discuss programmed grammars. More specifically, we discuss their nondeterministic behavior and its reduction. We prove that for every programmed grammar, there exists an equivalent programmed grammar where only a single rule has more than one successor. Furthermore, we establish an infinite hierarchy of language families resulting from the cardinality of successor sets. Open problem areas are formulated in the conclusion of the paper.

1 Introduction

A *programmed grammar* (see [3, 12]), G, is a context-free grammar, in which a set of rules—called *successors*—is attached to each rule. G can apply a rule r in the following way. If the left-hand side of r occurs in the sentential form under scan, G rewrites the left-hand side of r to its right-hand side, and during the next derivation step, it has to apply a rule from the set attached to r.

Since their introduction, programmed grammars have represented a vividly studied area of formal language theory, as demonstrated by several recent studies, such as [1, 2, 4–9]. Although this theory has established their fundamental properties (see [3, 10, 13] for a highlight of crucially important results), the precise role of nondeterminism in programmed grammars has not been studied to its fullness. In [1] and [2], it is proved that (a) if we require every rule in a programmed grammar to have at most one successor, then we can generate only finite languages, and (b) any programmed grammar can be converted to an equivalent programmed grammar with every rule having at most two successors. Also, other related results are proved in there. However, to our knowledge, there has been no study in

terms of (i) the maximum needed number of rules with more than one successor, and (ii) the impact of the overall number of successors in rules with two or more successors to the generative power of programmed grammars. The aim of this paper is, therefore, to fill this gap.

To study (i), we introduce a new normal form for programmed grammars, called the *one-ND rule normal form* (ND stands for *n*on*d*eterministic), where no more than one rule has more than one successor. We prove that every programmed grammar can be converted to this form.

To study (ii), we propose a new measure, called the *overall nondeterminism* of a programmed grammar, as the sum of all successors of rules with two or more successors. Then, informally speaking, we show that this measure gives rise to an infinite hierarchy of language families. More precisely, let n denote the overall sum of successors of rules with two or more successors. We prove that programmed grammars with $n + 1$ such nondeterministic choices can generate more languages than programmed grammars with only n choices.

It should be noted that some related measures of programmed grammars and other regulated grammars are discussed in Section 4.3 of [3] and in [5, 7]. For example, on page 192 in [3], it is pointed out that the more matrices in matrix grammars we have, the more power we have. In [5, 7], the needed number of nonterminals to sustain the power of programmed grammars is discussed. As obvious, our result can be thus seen as a continuation of these studies.

The paper is organized as follows. First, Section 2 gives all the necessary terminology. Then, Section 3 rigorously establishes the two results, (i) and (ii), sketched above. Finally, Section 4 formulates some open problem areas.

2 Preliminaries and Definitions

This paper assumes that the reader is familiar with the theory of formal languages (see [11]), including the theory of regulated rewriting (see [3]). For a set, Q, $\mathrm{card}(Q)$ denotes the cardinality of Q, and 2^Q denotes the power set of Q. 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. For $w \in V^*$, $|w|$ denotes the length of w.

Definition 1. A *programmed grammar* (see [3, 12]) is a quintuple, $G = (N, T, S, \Psi, P)$, where N is an alphabet of *nonterminals*, T is an alphabet of *terminals* ($N \cap T = \emptyset$), $S \in N$ is the *start symbol*, Ψ is an alphabet of *rule labels*, and $P \subseteq \Psi \times N \times (N \cup T)^* \times 2^\Psi$ is a finite relation such that $\mathrm{card}(\Psi) = \mathrm{card}(P)$, and for $(r, A, x, \sigma_r), (q, B, y, \sigma_q) \in P$, if $(r, A, x, \sigma_r) \neq (q, B, y, \sigma_q)$, then $r \neq q$.

Elements of P are called *rules*. Instead of $(r, A, x, \sigma_r) \in P$, we write

$\lfloor r\colon A \to x, \sigma_r \rfloor \in P$ throughout this paper. For $\lfloor r\colon A \to x, \sigma_r \rfloor \in P$, A is referred to as the *left-hand side* of r, and x is referred to as the *right-hand side* of r. Let $V = N \cup T$ be the *total alphabet*. G is *propagating* if and only if every $\lfloor r\colon A \to x, \sigma_r \rfloor \in P$ satisfies $x \in V^+$. Rules of the form $\lfloor r\colon A \to \varepsilon, \sigma_r \rfloor$ are called *erasing rules*.

The relation of a *direct derivation*, symbolically denoted by $\Rightarrow$, is defined over $V^* \times \Psi$ as follows: for $(x_1, r), (x_2, s) \in V^* \times \Psi$, $(x_1, r) \Rightarrow (x_2, s)$ (or $(x_1, r) \Rightarrow_G (x_2, s)$, if there is a danger of confusion) if and only if $x_1 = yAz$, $x_2 = ywz$, $\lfloor r\colon A \to w, \sigma_r \rfloor \in P$, and $s \in \sigma_r$.

Let $\lfloor r\colon A \to w, \sigma_r \rfloor \in P$. Then, σ_r is called the *success field* of r. Observe that, due to our definition of the relation of a direct derivation, if $\sigma_r = \emptyset$, then r is never applicable. Therefore, we assume that $\sigma_r \neq \emptyset$, for all $\lfloor r\colon A \to w, \sigma_r \rfloor \in P$, throughout the rest of our paper[1]. Let $\Rightarrow^n$, $\Rightarrow^*$, and $\Rightarrow^+$ denote the nth power of $\Rightarrow$, for some $n \geq 0$, the reflexive-transitive closure of $\Rightarrow$, and the transitive closure of $\Rightarrow$, respectively. Let $(S, r) \Rightarrow^* (w, s)$, where $r, s \in \Psi$ and $w \in V^*$. The *language generated by* G is denoted by $L(G)$ and defined as $L(G) = \{w \in T^* \mid (S, r) \Rightarrow^* (w, s)$, for some $r, s \in \Psi\}$. ■

Next, we define a new normal form and a new measure for programmed grammars. Then, we illustrate the defined notions by an example.

Definition 2. Let $G = (N, T, S, \Psi, P)$ be a programmed grammar. G is in the *one-ND rule normal form* (ND stands for *n*on*d*eterministic) if at most one $\lfloor r\colon A \to x, \sigma_r \rfloor \in P$ satisfies $\mathrm{card}(\sigma_r) \geq 1$ and every other $\lfloor s\colon B \to y, \sigma_s \rfloor \in P$ satisfies $\mathrm{card}(\sigma_s) \leq 1$. ■

Definition 3. Let $G = (N, T, S, \Psi, P)$ be a programmed grammar. For each $\lfloor r\colon A \to x, \sigma_r \rfloor \in P$, let $\zeta(r)$ be defined as:

$$\zeta(r) = \begin{cases} \mathrm{card}(\sigma_r) & \text{if } \mathrm{card}(\sigma_r) \geq 2 \\ 0 & \text{otherwise.} \end{cases}$$

The *overall nondeterminism of* G is denoted by $\mathrm{ond}(G)$ and defined as $\mathrm{ond}(G) = \sum_{r \in \Psi} \zeta(r)$. ■

By $\mathbf{P}$, we denote the family of languages generated by programmed grammars. By ${}_1\mathbf{P}$, we denote the family of languages generated by programmed grammars in the one-ND rule normal form.

Definition 4. Let $X \in \{\mathbf{P}, {}_1\mathbf{P}\}$ and $L \in X$. Then, we define

[1] Recall that such a definition is not unusual in the literature (see, for instance, [8]).

- $\mathrm{ond}(X, L) = \min\{\mathrm{ond}(G) \mid G \text{ is in the form generating } X,\ L = L(G)\}$,
- $\mathbf{OND}(X, n) = \{M \in X \mid \mathrm{ond}(X, M) \leq n\}$. ■

Example 1. Consider the language $K = \{a^n b^n c^n \mid n \geq 1\}$. This non-context-free language is generated by the programmed grammar $G = (N, T, S, \Psi, P)$, where $N = \{S, A, B, C\}$, $T = \{a, b, c\}$, and P contains the seven rules

$\lfloor r_1 \colon S \to ABC, \{r_2, r_5\} \rfloor$, $\lfloor r_5 \colon A \to a, \{r_6\} \rfloor$,

$\lfloor r_2 \colon A \to aA, \{r_3\} \rfloor$, $\lfloor r_6 \colon B \to b, \{r_7\} \rfloor$,

$\lfloor r_3 \colon B \to bB, \{r_4\} \rfloor$, $\lfloor r_7 \colon C \to c, \{r_7\} \rfloor$.

$\lfloor r_4 \colon C \to cC, \{r_2, r_5\} \rfloor$,

For example, $aabbcc$ is generated by $(S, r_1) \Rightarrow (ABC, r_2) \Rightarrow (aABC, r_3) \Rightarrow (aAbBC, r_4) \Rightarrow (aAbBcC, r_5) \Rightarrow (aabBcC, r_6) \Rightarrow (aabbcC, r_7) \Rightarrow (aabbcc, r_7)$.

Since the only nondeterministic rules are r_1 and r_2, each containing two rules in their successor sets, the overall nondeterminism of G is $\mathrm{ond}(G) = 4$. Therefore, $K \in \mathbf{OND}(\mathbf{P}, 4)$.

Because of r_1 and $_2$, G is not in the one-ND rule normal form. However, consider the programmed grammar $H = (N, T, S, \Psi', P')$, where P' contains the eight rules

$\lfloor r_0 \colon S \to abc, \{r_0\} \rfloor$, $\lfloor r_4 \colon C \to cC, \{r_2, r_5\} \rfloor$,

$\lfloor r_1 \colon S \to ABC, \{r_2\} \rfloor$, $\lfloor r_5 \colon A \to a, \{r_6\} \rfloor$,

$\lfloor r_2 \colon A \to aA, \{r_3\} \rfloor$, $\lfloor r_6 \colon B \to b, \{r_7\} \rfloor$,

$\lfloor r_3 \colon B \to bB, \{r_4\} \rfloor$, $\lfloor r_7 \colon C \to c, \{r_7\} \rfloor$.

Clearly, $L(H) = K$ and H is in the one-ND rule normal form. Observe that $\mathrm{ond}(H) = 2$, so $K \in \mathbf{OND}(\mathbf{P}, 2)$. Finally, since H is in the one-ND rule normal form, $K \in \mathbf{OND}(_1\mathbf{P}, 2)$. In the next section, we show that $\mathbf{OND}(\mathbf{P}, n) = \mathbf{OND}(_1\mathbf{P}, n)$. ■

3 Results

The following algorithm converts any programmed grammar, G, to an equivalent programmed grammar, G', in the one-ND rule normal form. To give an insight

into this conversion, we first explain the underlying idea behind it. First, we introduce the only nondeterministic rule of G', $\lfloor X : \# \to \varepsilon, \sigma_X \rfloor$. Obviously, each nondeterministic choice of some rule $\lfloor r : A \to x, \{s_1, s_2, \ldots, s_n\} \rfloor$ of G has to be simulated using X. To ensure proper simulation, we have to satisfy that (i) one of s_i is applied after r, and (ii) no other rules can be applied after r.

To satisfy both of these requirements, we introduce a special nonterminal symbol, $\langle r \rangle$, for each rule r of G. These symbols are used to encode the information about the last applied rule in a derivation. Then, for each successor of r, s_i, we introduce the following sequence of rules:

- $\lfloor r : A \to \langle r \rangle \#, \{X\} \rfloor$ to preserve the information that r is the currently simulated rule,
- X to make a nondeterministic choice of the successor of r, and
- $\lfloor \langle r \rhd s_i \rangle : \langle r \rangle \to x, \{s_i\} \rfloor$ to simulate r and continue with s_i.

Note that if X chooses some $\langle p \rhd q \rangle$ with $p \neq r$ instead, the derivation gets blocked because $\langle p \rangle$ is not present in the current sentential form.

Algorithm 1. Conversion of any programmed grammar to the one-ND rule normal form.

Input: A programmed grammar, $G = (N, T, S, \Psi, P)$.

Output: A programmed grammar in the one-ND rule normal form, $G' = (N', T, S', \Psi', P')$, such that $L(G') = L(G)$.

Method: Initially, set:

$N' = N \cup \{\#\} \cup \{\langle r \rangle \mid r \in \Psi\}$;
$\Psi' = \Psi \cup \{X\}$ with X being a new unique symbol;
$P' = \{\lfloor r : A \to x, \sigma_r \rfloor \mid \lfloor r : A \to x, \sigma_r \rfloor \in P, \mathrm{card}(\sigma_r) = 1\} \cup \{\lfloor X : \# \to \varepsilon, \sigma_X \rfloor\}$ with σ_X being initially set to $\emptyset$.

Now, for each $\lfloor r : A \to \omega, \sigma_r \rfloor \in P$ satisfying $\mathrm{card}(\sigma_r) > 1$, apply the following two steps:

(1) add $\lfloor r : A \to \langle r \rangle \#, \{X\} \rfloor$ to P';
(2) for each $q \in \sigma_r$, add $\lfloor \langle r \rhd q \rangle : \langle r \rangle \to \omega, \{q\} \rfloor$ to P', $\langle r \rhd q \rangle$ to Ψ', and $\langle r \rhd q \rangle$ to σ_X. ■

Lemma 1. *Algorithm 1 is correct.*

Proof. Clearly, the algorithm always halts and G' is in the one-ND rule normal form. To establish $L(G) = L(G')$, we first prove $L(G) \subseteq L(G')$ by showing how derivations of G are simulated by G', and then we prove $L(G') \subseteq L(G)$ by showing how every $w \in L(G')$ can be generated by G.

First, we introduce some notions used later in the proof. Set $V = N \cup T$, $V' = N' \cup T$, and $\bar{N} = \{\langle r \rangle \mid r \in \Psi\}$. For $\lfloor p\colon A \to x, \sigma_p \rfloor \in P'$, let $\mathrm{lhs}(p) = A$ and $\mathrm{rhs}(p) = x$.

We say that a rule labeled by r is *used* in a derivation, if the derivation can be expressed as $(u, p) \Rightarrow^* (v, r) \Rightarrow^+ (w, q)$, and for two labels q, r, we say that q *follows* r in a derivation, if the derivation can be expressed as $(u_1, p_1) \Rightarrow^* (v, r) \Rightarrow (w, q) \Rightarrow^* (u_2, p_2)$.

Claim 1. *Let $(S, s) \Rightarrow_G^m (w, q)$, where $s, q \in \Psi$, $w \in V^*$, for some $m \geq 0$. Then, $(S, s) \Rightarrow_{G'}^* (w, q)$.*

Proof. This claim is established by induction on m, $m \geq 0$.

Basis. Let $m = 0$. Then, for $(S, s) \Rightarrow_G^0 (S, s)$, where $s \in \Psi$, there is $(S, s) \Rightarrow_{G'}^0 (S, s)$, so the basis holds.

Induction Hypothesis. Suppose that the claim holds for all derivations of length l or less, where $l \leq m$, for some $m \geq 0$.

Induction Step. Consider any derivation of the form $(S, s) \Rightarrow_G^{m+1} (w, q)$, where $w \in V^*$ and $s, q \in \Psi$. Since $m + 1 \geq 1$, this derivation can be expressed as $(S, s) \Rightarrow_G^m (u, p) \Rightarrow_G (w, q)$, where $u \in V^*$, $p \in \Psi$. By the induction hypothesis, $(S, s) \Rightarrow_{G'}^* (u, p)$.

Now, we consider two possible cases, (i) and (ii), based on whether $\lfloor p\colon A \to x, \sigma_p \rfloor \in P$ satisfies $\mathrm{card}(\sigma_p) = 1$ or $\mathrm{card}(\sigma_p) > 1$:

(i) Let $\mathrm{card}(\sigma_p) = 1$. Then, $\lfloor p\colon A \to x, \sigma_p \rfloor \in P'$ by the initialization part of the algorithm, so the induction step is completed for (i).

(ii) Let $\mathrm{card}(\sigma_p) > 1$. Then, P' contains the following three rules:

- $\lfloor p\colon A \to \langle p \rangle \#, \{X\} \rfloor$, created in (1),
- $\lfloor X\colon \# \to \varepsilon, \sigma_X \rfloor$, created in the initialization part of the algorithm, and
- $\lfloor \langle p \triangleright q \rangle\colon \langle p \rangle \to x, \{q\} \rfloor$, created in (2) from q, such that $\langle p \triangleright q \rangle \in \sigma_X$.

Based on these rules, $(u, p) \Rightarrow_{G'}^3 (w, q)$, so the induction step is completed for (ii). □

Now, we establish two claims which we use later in the proof of $L(G') \subseteq L(G)$.

Claim 2. *Let* $(S, s) \Rightarrow^*_{G'} (w, q)$, *where* $s, q \in \Psi'$ *and* $w \in V'^*$. *Then, either* $w \in V^*$, *or* w *can be expressed as* $u\langle r\rangle\#v$ *or* $u\langle r\rangle v$, *where* $u, v \in V^*$ *and* $\langle r\rangle \in \bar{N}$.

Proof. Observe that only the rules created in (1) have symbols from $N' - N$ on their right-hand side. These rules have to be followed by X, which have to be followed by some rule created in (2). As X erases $\#$ and rules created in (2) are rewriting symbols from $\bar{N}$ to some $\omega \in V^*$, each string derived from S can be expressed as u, $u\langle r\rangle\#v$, or $u\langle r\rangle v$, where $u, v \in V^*$ and $\langle r\rangle \in \bar{N}$. □

Claim 3. *Let* $(S, s) \Rightarrow^*_{G'} (w, q)$, *where* $s, q \in \Psi'$, *and* $w \in V'^*$. *If* $w \in V^*$, *then* $q \in \Psi$.

Proof. Let $w \in V^*$ and assume—for the purpose of contradiction—that $q \in \Psi' - \Psi$. Then, q is either X, or it is created in (2). As X is only in the success field of some $\lfloor r\colon A \to \langle r\rangle\#, \{X\}\rfloor \in P'$, created in (1), and labels created in (2) are only in the success field of X, the derivation of (w, q) can be expressed in one of the following two forms:

- $(S, s) \Rightarrow^*_{G'} (w_1 A w_2, r) \Rightarrow_{G'} (w_1\langle r\rangle\# w_2, X)$, or
- $(S, s) \Rightarrow^*_{G'} (w_1 A w_2, r) \Rightarrow_{G'} (w_1\langle r\rangle\# w_2, X) \Rightarrow_{G'} (w_1\langle r\rangle w_2, \langle r \rhd q\rangle)$,

where $w_1, w_2 \in V^*$, and $\langle r \rhd q\rangle \in \sigma_X$. Note that in both forms, w would have to contain some $\langle r\rangle \in \bar{N}$, contradicting $w \in V^*$. Thus, $q' \in \Psi$, so the claim holds. □

The following claim shows that for each $w \in V^*$ derived in G', there is a derivation of w in G.

Claim 4. *Let* $(S, s) \Rightarrow^m_{G'} (w, q)$, *where* $s, q \in \Psi$, $w \in V^*$, *for some* $m \geq 0$. *Then,* $(S, s) \Rightarrow^*_G (w, q)$.

Proof. This claim is established by induction on m, $m \geq 0$.

Basis. Let $m = 0$. Then, for $(S, s) \Rightarrow^0_{G'} (S, s)$, where $s \in \Psi$, there is $(S, s) \Rightarrow^0_G (S, s)$, so the basis holds.

Induction Hypothesis. Suppose that the claim holds for all derivations of length l or less, where $l \leq m$, for some $m \geq 0$.

Induction Step. Consider any derivation of the form $(S, s) \Rightarrow^{m+1}_{G'} (w, q)$, where $w \in V^*$ and $s, q \in \Psi$. Since $m + 1 \geq 1$, this derivation can be expressed as

$(S, s) \Rightarrow_{G'}^{m} (u, p) \Rightarrow_{G'} (w, q)$, where $u \in V'^*$, $p \in \Psi'$. Now, we consider all possible forms of $(u, p) \Rightarrow_{G'} (w, q)$, covered by the next two cases:

(i) Let $u \in V^*$. Then, by Claim 3, $p \in \Psi$. Therefore, $\lfloor p\colon A \to \omega, \{q\} \rfloor \in P'$ is one of the rules created in the initialization part of the algorithm, so it is also in P. Thus, $(u, p) \Rightarrow_G (w, q)$. By the induction hypothesis, $(S, s) \Rightarrow_G^* (u, p)$, so the induction step is completed for (i).

(ii) Let $u = u_1\langle r\rangle u_2$, where $u_1, u_2 \in V^*$ and $\langle r\rangle \in \bar{N}$ for some $r \in \Psi$. Observe that $\lfloor r\colon A \to \langle r\rangle\#, \{X\} \rfloor \in P'$, created in (1), is the only rule with $\langle r\rangle$ on its right-hand side. Furthermore, observe that X has to follow r in the derivation. As $\mathrm{rhs}(r)$ contains $\#$ and only the rule labeled by X erases $\#$, X has to be used after the last occurrence of r in the derivation.

Now, consider all the labes in σ_X. These labels belong to the rules created in (2). Only these rules are rewriting the symbols from $\bar{N}$ to some $\omega \in V^*$, so they have to be used after (u, p) to satisfy $(u, p) \Rightarrow_{G'} (w, q)$. As p is followed by q, $p = \langle r \rhd q\rangle$, so the derivation can be expressed as

$$\begin{array}{lll} (S, s) & \Rightarrow_{G'}^{m-3} & (u_1 A u_2, r) \\ & \Rightarrow_{G'} & (u_1\langle r\rangle\# u_2, X) \\ & \Rightarrow_{G'} & (u_1\langle r\rangle u_2, \langle r \rhd q\rangle) \\ & \Rightarrow_{G'} & (w, q). \end{array}$$

Observe that $\lfloor r\colon A \to \langle r\rangle\#, \{X\} \rfloor \in P'$ is created in (1) from some $\lfloor r\colon A \to \omega, \sigma_r \rfloor \in P$ with $q \in \sigma_r$. Thus, $(u_1 A u_2, r) \Rightarrow_G (w, q)$. Clearly, $u_1 A u_2 \in V^*$. Therefore, by the induction hypothesis, $(S, s) \Rightarrow_G^* (u_1 A u_2, r)$, so the induction step is completed for (ii).

These cases cover only two of the three possible forms of u (see Claim 2). However, if $u = u_1\langle r\rangle\# u_2$, then $p = X$. As σ_X contains only rules created in (2), q would have to be in $\Psi' - \Psi$, contradicting $q \in \Psi$. Thus, cases (i) and (ii) cover all possible forms of $(u, p) \Rightarrow_{G'} (w, q)$, so the claim holds. □

To establish $L(G) = L(G')$, it suffices to show the following two statements:

- by Claim 1, for each $(S, s) \Rightarrow_G^* (w, q)$, where $s, q \in \Psi$ and $w \in T^*$, there is $(S, s) \Rightarrow_{G'}^* (w, q)$, so $L(G) \subseteq L(G')$;
- let $(S, s) \Rightarrow_{G'}^* (w, q)$, where $s, q \in \Psi'$ and $w \in T^*$. As $w \neq S$, s is used in the derivation, and so $\mathrm{lhs}(s) = S$. Observe that only rules with labels from Ψ have S on their left-hand side, so $s \in \Psi$. As $w \in T^*$, $q \in \Psi$ by Claim 3. Then, by Claim 4, $(S, s) \Rightarrow_G^* (w, q)$, so $L(G') \subseteq L(G)$.

As $L(G) \subseteq L(G')$ and $L(G') \subseteq L(G)$, $L(G) = L(G')$, so the lemma holds. □

The following theorem represents the first main achievement of this paper.

Theorem 1. *For any programmed grammar, G, there is a programmed grammar in the one-ND rule normal form, G', such that $L(G') = L(G)$.*

Proof. This theorem follows from Algorithm 1 and Lemma 1. □

Now, we study the impact of the overall number of successors in rules with two or more successors to the generative power of programmed grammars. First, however, we introduce some notions. Let $G = (N, T, S, \Psi, P)$ be a programmed grammar in the one-ND rule normal form. Set $V = N \cup T$, and for each $\lfloor r\colon A \to x, \sigma_r \rfloor \in P$, let $\sigma(r) = \sigma_r$. For $u \in V^*$ and $W \subseteq V$, let $\mathrm{occur}(u, W)$ denote the number of occurrences of symbols from W in u.

Let $s = (r_1, r_2, \ldots, r_k)$, where $r_i \in \Psi$, for all $i = 1, 2, \ldots, k$, be a sequence of labels. We say that s is *deterministic* if $\sigma(r_{i-1}) = \{r_i\}$, for $i = 2, 3, \ldots, k$. Each sequence of labels of the form $(r_1, r_2, \ldots, r_j)$, where $j \leq k$, is called a *prefix* of s.

We say that s *generates* a, where $a \in V$, if some $\lfloor r_i\colon A \to x, \sigma \rfloor \in P$ satisfies $x = uav$, where $u, v \in V^*$, for some i, $1 \leq i \leq k$. We say that a derivation *contains* s if it can be expressed as $(u, p) \Rightarrow^* (w_1, r_1) \Rightarrow (w_2, r_2) \Rightarrow \cdots \Rightarrow (w_k, r_k) \Rightarrow^* (v, q)$, where $u, v, w_i \in V^*$, $p, q \in \Psi$, for all $i = 1, 2, \ldots, k$. Let $\Rightarrow_{[s]}$ be a binary relation defined over V^* as follows: for $u, v \in V^*$, $u \Rightarrow_{[s]} v$ if and only if there is a derivation $(w_1, r_1) \Rightarrow (w_2, r_2) \Rightarrow \cdots \Rightarrow (w_k, r_k) \Rightarrow (v, p)$ such that $w_1 = u$.

Let Q_r be the set of all deterministic sequences beginning with $r \in P$ and let $\leq$ be a binary relation over Q_r defined as $s \leq t$ if and only if s is a prefix of t. It is easy to see that $\leq$ is reflexive, antisymmetric, transitive, and total (as all the sequences are deterministic and starting with the same rule). As there is a least element for every nonempty subset of Q_r, $(Q_r, \leq)$ is a well-ordered set.

We say that Q_r *generates* $a \in V$ if and only if there is some $s \in Q_r$ such that s generates a. We say that Q_r *reduces nonterminals* if there is some $s \in Q_r$ such that for each $t \geq s$ and for each $u, v \in V^*$, if $u \Rightarrow_{[t]} v$, then $\mathrm{occur}(u, N) > \mathrm{occur}(v, N)$.

Lemma 2. $\mathbf{OND}(\mathbf{P}, n) = \mathbf{OND}({}_1\mathbf{P}, n)$

Proof. Let $G = (N, T, S, \Psi, P)$ be a programmed grammar. Then, by Algorithm 1 and Lemma 1, we can construct a programmed grammar in the one-ND rule form, $G' = (N', T, S', \Psi', P')$, such that $L(G) = L(G')$ and

$\lfloor r\colon A \to x, \sigma_r \rfloor \in P' - \{\lfloor X\colon \# \to \varepsilon, \sigma_X \rfloor\}$ satisfies $\mathrm{card}(\sigma_r) \leq 1$. Observe that for each $\lfloor r\colon A \to x, \sigma_r \rfloor \in P$ with $\mathrm{card}(\sigma_r) > 1$, there are $\mathrm{card}(\sigma_r)$ labels in σ_X created in (2) of Algorithm 1. As these are the only labels in σ_X, and all other rules in P' have at most one label in their success field, by Definition 3, $\mathrm{ond}(G') = \mathrm{ond}(G)$. Thus, $\mathbf{OND}(\mathbf{P}, n) \subseteq \mathbf{OND}({}_1\mathbf{P}, n)$. Obviously, $\mathbf{OND}({}_1\mathbf{P}, n) \subseteq \mathbf{OND}(\mathbf{P}, n)$, so the lemma holds. □

Lemma 3. *Let $G = (N, T, S, \Psi, P)$ be a programmed grammar in the one-ND rule normal form such that $L(G)$ is infinite. Then, there is exactly one $r \in \Psi$ such that $\mathrm{card}(\sigma(r)) > 1$.*

Proof. This lemma follows from Definition 2 in Section 2 and from Lemma 8 in [2], which says that programmed grammars with every rule having at most one successor generate only finite languages. □

Lemma 4. *Let $G = (N, T, S, \Psi, P)$ be a programmed grammar in the one-ND rule normal form such that $L(G)$ is infinite, and let r_x denote the only rule satisfying $\mathrm{card}(\sigma(r_x)) > 1$. Then, there are $p, q \in \sigma(r_x)$ such that Q_p reduces nonterminals and Q_q does not reduce nonterminals.*

Proof. Observe that there is a finite number of $u \in V^*$ that can be derived without r_x being used in their derivation. Furthemore, r_x has to be used arbitrary many times to derive a string of arbitrary length (see Lemma 3).

Now, we prove by contradiction that there is at least one $r \in \sigma(r_x)$ such that Q_r reduces nonterminals. Assume—for the purpose of contradiction—that each Q_r, where $r \in \sigma(r_x)$, does not reduce nonterminals. Then, for sufficiently large k, $(u, r_x) \Rightarrow^k (w, q)$ implies $w \notin T^*$. Therefore, $L(G)$ would be finite, which leads to a contradiction. Thus, there is at least one $r \in \sigma(r_x)$ such that Q_r reduces nonterminals.

Now, we prove by contradiction that there is at least one $r \in \sigma(r_x)$ such that Q_r does not reduce nonterminals. Assume—for the purpose of contradiction—that each Q_r, where $r \in \sigma(r_x)$, reduces nonterminals. Then, there exists some $k \geq 0$ such that each $(u, r_x) \Rightarrow^+ (w, q)$, where $w \in T^*$, implies $|w| \leq k|u|$. As there is a finite number of such u that can be derived from S without using r_x, $L(G)$ is finite, which leads to a contradiction. Thus, the lemma holds. □

Lemma 5. $\mathbf{OND}({}_1\mathbf{P}, n) \subset \mathbf{OND}({}_1\mathbf{P}, n+1)$

Proof. Let L_n be a language over $\Sigma = \{a_1, a_2, \ldots a_n\}$, defined as

$$L_n = \bigcup_{i=1}^{n} \{a_i\}^+.$$

We show that $\mathrm{ond}({}_1\mathbf{P}, L_n) = n + 1$.

First, observe that L_n is generated by the propagating programmed grammar

$$G = \big(\{S\}, \{a_1, a_2, \ldots, a_n\}, S, \{r_S, r_1, r_2, \ldots, r_n\}, P'\big)$$

with

$$P' = \big\{ \lfloor r_S \colon S \to SS, \{r_S, r_1, r_2, \ldots, r_n\} \rfloor \big\} \cup \big\{ \lfloor r_i \colon S \to a_i, \{r_i\} \rfloor \mid 1 \leq i \leq n \big\}.$$

As the cardinality of the success field of r_S is $n + 1$, $\mathrm{ond}({}_1\mathbf{P}, L_n) \leq n + 1$.

Now, we show that every programmed grammar in the one-ND rule normal form generating L_n requires at least one rule with $n + 1$ labels in its success field. Let $G' = (N, T, S, \Psi, P)$, where $T = \{a_1, a_2, \ldots, a_n\}$, be a programmed grammar in the one-ND rule normal form such that $L(G') = L_n$. As L_n is infinite, by Lemma 3, there is exactly one $\lfloor r \colon A \to x, \sigma_r \rfloor \in P$ satisfying $\mathrm{card}(\sigma_r) > 1$. Let r_x denote this rule.

First, we prove that there is at least one $r_a \in \sigma(r_x)$ for each $a \in T$. Then, we show that there has to be at least one additional rule in $\sigma(r_x)$ to generate all the strings in L_n.

Claim 5. *For each $a \in T$, there has to be $r \in \sigma(r_x)$ such that Q_r generates a and Q_r does not generate any $b \in T$, $b \neq a$.*

Proof. For the purpose of contradiction, assume the contrary—that is, assume that there is $a \in T$ such that each Q_r generating a generates also some $b \in T$, $b \neq a$. Let s denote the shortest sequence in Q_r generating both a and b. Observe that there is no string in L_n for which there is a derivation in G' containing s, or some $t \geq s$ (such a string would have to contain both a and b). As all the sequences in Q_r are deterministic, any prefix of s could be contained at most once in any successful derivation. As there is a limited number of such prefixes, it would be impossible to derive a^m for arbitrary m, contradicting $L(G') = L_n$, so the claim holds. □

By Claim 5, there has to be $r \in \sigma(r_x)$ for each $a \in T$ such that Q_r generates only a. Let $Q(a)$ denote such Q_r. Now, we show that there is at least one additional rule in $\sigma(r_x)$. Consider the following two cases, based on whether each $Q(a)$ reduces nonterminals or not:

(i) Each $Q(a)$ does not reduce nonterminals. Since L_n is infinite, by Lemma 4, there has to be at least one additional $p \in \sigma(r_x)$ such that Q_p reduces nonterminals.

(ii) At least one $Q(a)$ reduces nonterminals. Now, assume—for the purpose of contradiction—that $\mathrm{card}(\sigma(r_x)) = n$. As only $Q(a)$ generates a, and it also reduces nonterminals, there is some $k \geq 0$ such that each $(u, r_x) \Rightarrow^+ (w, q)$, where $q \in \Psi$, $u \in V^*$, and $w \in \{a\}^*$, implies $|w| \leq k|u|$. As there is a limited number of such u that can be derived from S without using r_x, a^m cannot be derived for arbitrary m, which leads to a contradiction. Thus, there has to be an additional rule in $\sigma(r_x)$.

Observe that these two cases cover all possible $Q(a)$ for each $a \in T$. Therefore, $\mathrm{card}(\sigma(r_x)) \geq n+1$, which implies $\mathrm{ond}({}_1\mathbf{P}, L_n) \geq n+1$. Therefore, $L_n \notin \mathbf{OND}({}_1\mathbf{P}, n)$. Since $\mathrm{ond}({}_1\mathbf{P}, L_n) \leq n+1$ implies $L_n \in \mathbf{OND}({}_1\mathbf{P}, n+1)$, the lemma holds. □

The following theorem represents the second main achievement of this paper.

Theorem 2. $\mathbf{OND}(\mathbf{P}, n) \subset \mathbf{OND}(\mathbf{P}, n+1)$

Proof. This theorem follows Lemma 2 and Lemma 5. □

4 Concluding Remarks

In this concluding section of our paper, we formulate some open problem areas. First, consider *programmed grammars with appearance checking* (see [3]). Recall that in these grammars, another set of rules—called the *failure field*—is attached to every rule of the underlying context-free grammar. Then, a rule like this can be either applied in the same way as in a programmed grammar, in which case we pass to a rule from its success field, or, if this rule is not applicable, then the sentential form remains unchanged and we pass to a rule from its failure field. Do the achieved results also hold in terms of programmed grammars with appearance checking?

Second, reconsider Algorithm 1. Observe that it introduces erasing rules to G', even if the input grammar, G, is propagating. Can we modify this algorithm so that when G is propagating, then so is G'? Furthermore, do Theorems 1 and 2 hold in case of propagating programmed grammars? Observe that the argument Lemma 5 is based on holds in terms of propagating programmed grammars, too.

Acknowledgments

This work was supported by the MSM0021630528 and FIT-S-11-2 grants.

References

[1] M. Barbaiani, C. Bibire, J. Dassow, A. Delaney, S. Fazekas, M. Ionescu, G. Liu, A. Lodhi, and B. Nagy. The power of programmed grammars with graphs from various classes. *Journal of Applied Mathematics & Computing*, 22(1–2):21–38, 2006.

[2] H. Bordihn and M. Holzer. Programmed grammars and their relation to the LBA problem. *Acta Informatica*, 43(4):223–242, 2006.

[3] J. Dassow and G. Păun. *Regulated Rewriting in Formal Language Theory*. Springer, New York, 1989.

[4] H. Fernau. Unconditional transfer in regulated rewriting. *Acta Informatica*, 34(11):837–857, 1997.

[5] H. Fernau. Nonterminal complexity of programmed grammars. *Theoretical Computer Science*, 296(2):225–251, 2003.

[6] H. Fernau. Programmed grammars with rule queues. *International Journal of Foundations of Computer Science*, 18(6):1209–1213, 2007.

[7] H. Fernau, R. Freund, M. Oswald, and K. Reinhardt. Refining the nonterminal complexity of graph-controlled, programmed, and matrix grammars. *Journal of Automata, Languages and Combinatorics*, 12(1–2):117–138, 2007.

[8] H. Fernau and F. Stephan. How powerful is unconditional transfer? — When UT meets AC. In *Developments in Language Theory*, pages 249–260, 1997.

[9] H. Fernau and F. Stephan. Characterizations of recursively enumerable sets by programmed grammars with unconditional transfer. *Journal of Automata, Languages and Combinatorics*, 4(2):117–152, 1999.

[10] C. Martín-Vide, V. Mitrana, and G. Păun, editors. *Formal Languages and Applications*, chapter 13, pages 249–274. Springer, Berlin, 2004.

[11] A. Meduna. *Automata and Languages: Theory and Applications*. Springer, London, 2000.

[12] D. J. Rosenkrantz. Programmed grammars and classes of formal languages. *Journal of the ACM*, 16(1):107–131, 1969.

[13] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages, Vol. 2: Linear Modeling: Background and Application*, chapter 3, pages 101–154. Springer, New York, 1997.