A NEW NORMAL FORM FOR PROGRAMMED GRAMMARS WITH APPEARANCE CHECKING

Lukáš Vrábel
Doctoral Degree Programme (2), FIT BUT
E-mail: xvrabe01@stud.fit.vutbr.cz

Supervised by: Alexander Meduna
E-mail: meduna@fit.vutbr.cz

Abstract: In the present paper, we discuss programmed grammars with appearance checking. We investigate the effect of the number of rules with more than one successor in success and/or failure field on generative power of the programmed grammars. We prove that for every programmed grammar, there is an equivalent programmed grammar where only a single rule has more than one successor in both success and failure fields.

Keywords: Programmed grammar with appearance checking, normal form, successor, nondeterminism

1 INTRODUCTION

In the formal language theory, programmed grammars have been thoroughly investigated (see [1, 2, 4–6, 8, 10] for recent studies). Although various properties of these grammars have been well established, the effect of rules with more than one successor has not been investigated to its full extent. In [1] and [2], it is proved that (a) to generate an infinite language, there has to be at least one rule with more than one successor, and (b) any programmed grammar can be converted to an equivalent programmed grammar with every rule having at most two successors. In [10], we proved that each programmed grammar can be converted to an equivalent programmed grammar with only one rule with more than one succesor, and in [8], we used this normal form to prove that we cannot limit overall number of successors.

In this paper, we continue the study of successor nondeterminism in programmed grammars. Normal forms significantly simplify the investigation of some properties of grammars and their corresponding languages. Furthermore, these forms make the application and implementation easier. Therefore, we extended the notions and normal form from [10] to programmed grammars with appearance checking, thus solving some of the open problems presented in [10] and [8]. Specifically, we extend the definition of *one-ND rule normal form* (ND stands for *n*on*d*eterministic), where at most one rule has more than one successor, to programmed grammars with appearance checking. Then we prove that every programmed grammar with appearance checking can be converted to this form.

2 PRELIMINARIES AND DEFINITIONS

This paper assumes that the reader is familiar with the theory of formal languages (see [7]), 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, 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.

Definition 1. A *programmed grammar with appearance checking* (see [3, 9]) is a quintuple, $G = (N, T, S, \Psi, P)$, where N is the alphabet of *nonterminals*, T is the alphabet of *terminals*, $N \cap T = \emptyset$, $S \in N$

is the *start symbol*, Ψ is the alphabet of *rule labels*, and $P \subseteq \Psi \times N \times (N \cup T)^* \times 2^\Psi \times 2^\Psi$ is a finite relation such that card$(\Psi) =$ card(P), and for $(r, A, x, \sigma_r, \phi_r), (q, B, y, \sigma_q, \phi_q) \in P$, if $(r, A, x, \sigma_r, \phi_r) \neq (q, B, y, \sigma_q, \phi_q)$, then $r \neq q$.

Elements of P are called *rules*. Instead of $(r, A, x, \sigma_r, \phi_r) \in P$, we write $\lfloor r \colon A \to x, \sigma_r, \phi_r \rfloor \in P$ throughout this paper. For $\lfloor r \colon A \to x, \sigma_r, \phi_r \rfloor \in P$, 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 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), where $\lfloor r \colon A \to w, \sigma_r, \phi_r \rfloor$, if and only if either

- $x_1 = yAz$, $x_2 = ywz$, and $s \in \sigma_r$, or
- $x_1 = x_2$, A does not occur in x_1, and $s \in \phi_r$.

Let $\lfloor r \colon A \to w, \sigma_r, \phi_r \rfloor \in P$. Then, σ_r is called the *success field* of r and ϕ_r is called the *failure field* of r. 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 $w \in V^*$. Then, (w, s) is called a *configuration*. The *language generated by* G is denoted by $L(G)$ and defined as $L(G) = \{w \in T^* \mid (S, r) \Rightarrow^* (w, s), \text{ for some } r, s \in \Psi\}$. ■

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 $\lfloor r \colon A \to x, \sigma_r, \phi_r \rfloor \in P$ satisfies card$(\sigma_r) \geq 1$ or card$(\phi_r) \geq 1$, and every other $\lfloor r \colon A \to x, \sigma_r \rfloor \in P$ satisfies card$(\sigma_r) \leq 1$ and card$(\phi_r) \leq 1$. ■

3 CONVERSION OF PROGRAMMED GRAMMARS INTO ONE-ND NORMAL 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:

$N' = N \cup \{\#, S'\} \cup \{\langle r_\sigma \rangle, \langle r_\phi \rangle \mid r \in \Psi\}$;
$\Psi' = \Psi \cup \{X\}$ with X being a new unique symbol;
$P' = \{\lfloor X \colon \# \to \#, \emptyset, \phi_X \rfloor\}$ with ϕ_X initially set to $\emptyset$.

Now, apply the following three steps:

(1) for each $\lfloor r \colon A \to \omega, \sigma_r, \phi_r \rfloor \in P$:
  (1.1) add $\lfloor r \colon A \to \omega, \{r_\sigma\}, \{r_\phi\} \rfloor$ to P',
  (1.2) add $\lfloor r_\sigma \colon \# \to \langle r_\sigma \rangle, \{X\}, \emptyset \rfloor$ to P', and r_σ to Ψ',
  (1.3) add $\lfloor r_\phi \colon \# \to \langle r_\phi \rangle, \{X\}, \emptyset \rfloor$ to P', and r_ϕ to Ψ',
  (1.4) for each $q \in \sigma_r$, add $\lfloor \langle r_\sigma \rhd q \rangle \colon \langle r_\sigma \rangle \to \#, \{q\}, \emptyset \rfloor$ to P', $\langle r_\sigma \rhd q \rangle$ to Ψ' and to ϕ_X;
  (1.5) for each $q \in \phi_r$, add $\lfloor \langle r_\phi \rhd q \rangle \colon \langle r_\phi \rangle \to \#, \{q\}, \emptyset \rfloor$ to P', $\langle r_\phi \rhd q \rangle$ to Ψ' and to ϕ_X;
(2) for each $\lfloor r \colon S \to \omega, \sigma_r, \phi_r \rfloor \in P'$:
  (2.1) add $\lfloor r_s \colon S' \to \#S, \{r\}, \emptyset \rfloor$ to P',

(2.2) add r_s to Ψ';

(3) for each $\lfloor\langle p \rhd q\rangle \colon \langle p\rangle \to \#, \{q\}, \emptyset\rfloor \in P'$ satisfying $\langle p \rhd q\rangle \in \phi_X$:

(3.1) add $\lfloor\langle p \rhd_\varepsilon q\rangle \colon \langle p\rangle \to \varepsilon, \{q\}, \emptyset\}\rfloor$ to P',

(3.2) add $\langle p \rhd_\varepsilon q\rangle$ to both Ψ' and ϕ_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 $s \in L(G')$ can be generated by G.

Set $V = N \cup T$ and $\bar{N} = N' - N$. Observe that all strings derived from S' in G' are of the form $\langle z\rangle u$, $\#u$, or u, where $\langle z\rangle \in \bar{N}$, $u \in V^*$.

Claim 1. *If $(u, r) \Rightarrow_G (w, q)$, then $(\#u, r) \Rightarrow^*_{G'} (\#w, q)$, where $u, w \in V^*$, $r, q \in \Psi$.*

Proof. Let $\lfloor r \colon A \to x, \sigma_r, \sigma_p\rfloor \in P$. Then, following rules are in P':

- $\lfloor r \colon A \to x, \{r_\sigma\}, \{r_\phi\}\rfloor$ created in (1.1),
- $\lfloor r_\sigma \colon \# \to \langle r_\sigma\rangle, \{X\}, \emptyset\rfloor$ created in (1.2),
- $\lfloor r_\phi \colon \# \to \langle r_\phi\rangle, \{X\}, \emptyset\rfloor$ created in (1.3),
- $\lfloor\langle r_\sigma \rhd q\rangle \colon \langle r_\sigma\rangle \to \#, \{q\}, \emptyset\rfloor$ created in (1.4),
- $\lfloor\langle r_\phi \rhd q\rangle \colon \langle r_\phi\rangle \to \#, \{q\}, \emptyset\rfloor$ created in (1.5).

There are two cases, (i) and (ii), based on whether u contain at least one occurence of A or not:

(i) Assume u contains at least one occurence of A. Then, $(\#u, r) \Rightarrow_{G'} (\#w, r_\sigma) \Rightarrow_{G'} (\langle r_\sigma\rangle w, X) \Rightarrow_{G'} (\langle r_\sigma\rangle w, \langle r_\sigma \rhd q\rangle) \Rightarrow_{G'} (\#w, q)$, so the lemma holds for this case.

(ii) Assume u does not contain any occurence of A. Then, $(\#u, r) \Rightarrow_{G'} (\#w, r_\phi) \Rightarrow_{G'} (\langle r_\phi\rangle w, X) \Rightarrow_{G'} (\langle r_\phi\rangle w, \langle r_\phi \rhd q\rangle) \Rightarrow_{G'} (\#w, q)$, so the lemma holds for this case.

Observe, that these two cases cover all possible forms of $(u, r) \Rightarrow_G (w, q)$. Thus, the lemma holds. $\square$

Due to the size constraint of this paper, the proofs of the following two claims are left to the reader.

Claim 2. *If $(S', \alpha) \Rightarrow^*_{G'} (\#w, q)$, where $\alpha, q \in \Psi'$, $w \in V^*$, then $(S', \alpha') \Rightarrow^*_{G'} (w, q')$, where $\alpha', q' \in \Psi'$.*

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

Claim 1 establishes the relation between the derivation step in G and its counterpart in G'. Claims 2 and 3 show the relation between $w \in V^*$ derived in G' from S', and its corresponding sentence form, $\#\langle z\rangle w$, containing the symbol used to preserve the information about the last applied rule.

The following claim demonstrates how derivations of G are simulated by G'.

Claim 4. *Let $(S, r) \Rightarrow^m_G (w, q)$, where $r, q \in \Psi$, $w \in V^*$, for some $m \geq 1$. Then, $(S', r_s) \Rightarrow^*_{G'} (\#w, q)$, where $r_s \in \Psi'$.*

Proof. This claim is established by induction on m, $m \geq 1$.

Basis. Let $m = 1$. Then, $(S,r) \Rightarrow_G (w,q)$ by some $r \in \Psi$. By Claim 1, $(\#S,r) \Rightarrow^*_{G'} (\#w,q)$. Since r has S on its left-hand side, $\lfloor r_s\colon S' \to \#\langle\emptyset\rangle S, \{r\}, \emptyset\rfloor \in P'$ by (2.1), so $(S',r_s) \Rightarrow_{G'} (\#S,r) \Rightarrow^*_{G'} (\#w,q)$. Thus, the basis holds.

Induction Hypothesis. Suppose that the claim holds for all derivations of length l or less, where $l \leq m$, for some $m \geq 1$.

Induction Step. Consider any derivation of the form $(S,r) \Rightarrow^{m+1}_G (w,q)$, where $w \in V^*$ and $r,q \in \Psi$. Since $m+1 \geq 1$, this derivation can be expressed as $(S,r) \Rightarrow^m_G (x,p) \Rightarrow_G (w,q)$, where $x \in V^*$, $p \in \Psi$. By the induction hypothesis, $(S',r_s) \Rightarrow^*_{G'} (\#x,p)$, and by Claim 1, $(\#x,p) \Rightarrow^*_{G'} (\#w,q)$. Thus, the claim holds. $\square$

Now, we show that for each derivation of $\#u$ in G', there is a derivation of u in G, which will be later used to prove $L(G') \subseteq L(G)$.

Claim 5. *If $(S',r_s) \Rightarrow^m_{G'} (\#u,q)$, for some $m \geq 1$, then $(S,r) \Rightarrow^*_G (u,q)$, where $r,q \in \Psi$, $r_s \in \Psi'$, and $u \in V^*$.*

Proof. This claim is established by induction on m, $m \geq 1$.

Basis. Let $m = 1$. Then, $(S',r_s) \Rightarrow_{G'} (\#S,r)$. As r_s is created in (2.1) from $r \in \Psi$, $(S,r) \Rightarrow^0_G (S,r)$, 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 $m \geq 1$.

Induction Step. Consider any $(S',r_s) \Rightarrow^{m+1}_{G'} (\#u,q)$, where $u \in V^*$ and $r_s,q \in \Psi$. Since $m+1 \geq 2$, this derivation can be expressed as

$$(S',r_s) \Rightarrow^m_{G'} (\langle z\rangle v, \langle z \rhd q\rangle) \Rightarrow_{G'} (\#u,q),$$

where $v \in V^*$, $\langle z\rangle \in \bar{N}$, and $\langle z \rhd q\rangle \in \Psi'$. Observe, that $\langle z \rhd q\rangle$ was created in (1.4) or (1.5) from some $\lfloor p\colon A \to \omega, \sigma_p, \phi_p\rfloor \in P$ and q is contained either σ_p or in ϕ_p. Therefore, $\langle z\rangle$ is either $\langle p_\sigma\rangle$ or $\langle p_\phi\rangle$. Recall, that only ϕ_X contains labels created in (1.4) or (1.5), so the derivation can be expressed as

$$(S',r_s) \Rightarrow^{m-1}_{G'} (\langle z\rangle v, X) \Rightarrow_{G'} (\langle z\rangle v, \langle z \rhd q\rangle) \Rightarrow_{G'} (\#u,q).$$

Note, that X is only in the success field of rules created in (1.2) or (1.3). Let p_σ and p_ϕ denote the rules created from p in (1.2) and (1.3), respectively. As $(\langle z\rangle v, \langle z \rhd q\rangle) \Rightarrow_{G'} (\#u,q)$, either p_σ or p_ϕ have to precede X in the derivation, so it can be expressed as

$$(S',r_s) \Rightarrow^{m-3}_{G'} (\#w,p) \Rightarrow_{G'} (\#v,p') \Rightarrow_{G'} (\langle z\rangle v, X) \Rightarrow_{G'} (\langle z\rangle v, \langle z \rhd q\rangle) \Rightarrow_{G'} (\#u,q),$$

where $w \in V^*$ and $p' \in \{p_\sigma, p_\phi\}$. Observe, that $(\#w,p) \Rightarrow_{G'} (\#v,p')$ holds due to the $\lfloor p\colon A \to \omega, \{p_\sigma\}, \{p_\phi\}\rfloor$ created in (1.1) from the same $\lfloor p\colon A \to \omega, \sigma_p, \phi_p\rfloor \in P$. Therefore, $(w,p) \Rightarrow_G (v,q)$. Note, that $p \in \Psi$ and $(S',r_s) \Rightarrow^{m-3}_{G'} (\#w,p)$. Then, by the induction hypothesis, $(S,r) \Rightarrow^*_G (w,p)$, so the claim holds. $\square$

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

- by Claim 4, for each $(S,r) \Rightarrow_G^* (u,q)$, where $r,q \in \Psi$, and $u \in T^*$, there is $(S',r_s) \Rightarrow_{G'}^* (\#u,q)$, where $r_s \in \Psi'$. Then, $(S',r') \Rightarrow_{G'}^* (u,q')$ by Claim 2, so $L(G) \subseteq L(G')$.
- by Claim 3, for each $(S',r') \Rightarrow_{G'}^* (u,q')$, where $r',q' \in \Psi'$ and $u \in T^*$, there is $(S',r_s) \Rightarrow_{G'}^* (\#u,q)$, where $r_s \in \Psi'$ and $q \in \Psi$. Then, $(S,r) \Rightarrow_G^* (u,q)$, where $r \in \Psi$, by Claim 5, 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 main achievement of this paper. The theorem follows from Algorithm 1 and Lemma 1.

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

4 CONCLUSION

In this section, we present some open problems. Observe, that Algorithm 1 introduces erasing rules to G', even if the input grammar is propagating. Can the algorithm be modified in such way, that when G is propagating, then so is G'?

Second, even the deterministic rules of G are processed by Algorithm 1, thus unnecessary increasing the descriptional complexity of G'. Can we modify the algorithm, so it will not introduce unnecessary symbols and rules to G'?

ACKNOWLEDGEMENT

This work was supported by the following grants: MŠMT FR271/2012/G1, BUT FIT FIT-S-11-2, EU CZ 1.05/1.1.00/02.0070, and CEZ MŠMT MSM 0021630528.

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, 1989.

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

[5] 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.

[6] 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.

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

[8] Alexander Meduna, Lukáš Vrábel, and Petr Zemek. On nondeterminism in programmed grammars. In *AFL*, 2011.

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

[10] Lukáš Vrábel. A new normal form for programmed grammars. In *In Proceedings of the 17th Conference STUDENT EEICT 2011*, 2011.