n-Right-Linear #-Rewriting Systems

Zbyněk Křivka[1], Alexander Meduna[1], and Jaromír Smrček[1]

Department of Information Systems
Faculty of Information Technology, Brno University of Technology
Božetěchova 1, Brno 612 66, Czech Republic
krivka@fit.vutbr.cz

Abstract. The present paper discusses #-rewriting systems, which represent simple language-defining devices that combine both automata and grammars. Indeed, like automata, they use finitely many states without any nonterminals; on the other hand, like grammars, they generate languages. The paper introduces n-right-linear #-rewriting systems and characterize the infinite hierarchy of language families defined by m-parallel n-right-linear simple matrix grammars. However, it also places some trivial restrictions on rewriting in these systems and demonstrates that under these restrictions, they generate only the family of right-linear languages. In its conclusion, this paper suggests some variants of #-rewriting systems.

1 Introduction

As one of its important topics, the descriptional complexity of rewriting systems investigates how a restriction placed on the rewriting process affects the language family defined by the systems (see Chapter 4 in [1] and Chapter 3 in [6]). In the present paper, we continue with this vivid trend of the descriptional complexity in terms of the recently introduced #-rewriting systems (see [3]). That is, we place a restriction on the number of rewriting positions during the process that yields the strings and demonstrate that this restriction give rise to an infinite hierarchy of language families.

We consider the #-rewriting systems of finite index (see [2], [3]), which can always rewrite any string at no more than k positions, for a positive integer k. Then, as their special cases, we introduce and study n-right-linear #-rewriting systems as the central topic of this paper. As their name indicates, these systems are underlain by rules that are similar to the right-linear grammatical rules. These systems characterize the infinite hierarchy of language families defined by m-parallel n-right-linear simple matrix grammars; however, under some trivial restrictions, they generate only the family of right-linear languages.

Regarding the applications, the #-rewriting systems discussed in this paper can be used to analyze and classify various texts into the achieved infinite hierarchy. Based on a classification of this kind, we can detect and remove undesirable pieces of information from the texts and, there by avoid its distribution.

2 Preliminaries

This paper assumes that the reader is familiar with formal language theory (see [4], [6]). For an alphabet V, V^* represents the free monoid generated by V under the operation of concatenation. The identity 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, and for $W \subseteq V$, $occur(w, W)$ denotes the number of occurrences of symbols from W in w. For $i = 1, 2, \dots, |w|$, $sym(w, i)$ denotes the i-th symbol of w; for instance, $sym(abcd, 3) = c$. For every $i \geq 0$, $suffix(w, i)$ is w's suffix of length i if $|w| \geq i$, and $suffix(w, i) = w$ if $i > |w|$. $suffixes(w) = \{suffix(w, j) \mid 0 \leq j \leq |w|\}$.

A *right-linear grammar* is a quadruple, $G = (V, T, P, S)$, where V is a total alphabet, $T \subseteq V$ is an alphabet of terminals, $S \in (V - T)$ is the start symbol, and P is a finite set of *rules* of the form $q\colon A \to v$, where $A \in (V - T)$, $v \in T^*(V - T) \cup T^*$ and q is an unique label of this rule. If $q\colon A \to v \in P$, $x, y \in V^*$, G makes a derivation step from xAy to xvy according to $q\colon A \to v$, symbolically written as $xAy \Rightarrow xvy\ [q]$ or, simply, $xAy \Rightarrow xvy$. In the standard manner, we define $\Rightarrow^m$, where $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. To express that G makes $x \Rightarrow^m y$, where $x, y \in V^*$, by using a sequence of rules $q_1, q_2, \dots, q_m$, we symbolically write $x \Rightarrow^m y\ [q_1 q_2 \dots q_m]$. The *language of* G, $L(G)$, is defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. A language, L, is *right-linear* if and only if $L = L(G)$, where G is a right-linear grammar. Let $\mathcal{L}(RLIN)$ denotes the family of right-linear languages.

For $p \in P$, $rhs(p)$ and $lhs(p)$ denotes the right-hand side and the left-hand side of rule p, respectively, $lab(p)$ denotes the label of rule p, and for $\overline{P} \subseteq P$, $lab(\overline{P})$ denotes the set of all labels of rules from $\overline{P}$. Instead of a rule, we frequently simply write its label in what follows for brevity.

For $m, n \geq 1$, an *m-parallel n-right-linear simple matrix grammar* (*m-Pn-G*, see [5]) is an $(mn + 3)$-tuple $G = (N_{11}, \dots, N_{1n}, \dots, N_{m1}, \dots, N_{mn}, T, S, P)$ where N_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$ are mutually disjoint *nonterminal alphabets*, T is a *terminal alphabet*, $S \notin N \cup T$ is the start symbol, where $N = N_{11} \cup \ldots \cup N_{mn}$, and P is a finite set of *matrix rules*.

A matrix rule can be in one of the following three forms:

(i) $[S \to X_{11} \dots X_{mn}]$, $X_{ij} \in N_{ij}$, $1 \leq i \leq m$, $1 \leq j \leq n$,
(ii) $[X_{i1} \to \alpha_{i1}, \dots, X_{in} \to \alpha_{in}]$, $X_{ij} \in N_{ij}$, $\alpha_{ij} \in T^*$, $1 \leq j \leq n$, for some i, $1 \leq i \leq m$, and
(iii) $[X_{i1} \to \alpha_{i1} Y_{i1}, \dots, X_{in} \to \alpha_{in} Y_{in}]$ $X_{ij}, Y_{ij} \in N_{ij}$, $\alpha_{ij} \in T^*$, $1 \leq j \leq n$, for some i, $1 \leq i \leq m$.

The derivation step for *m-Pn-G* is defined as follows:
For $x, y \in (N \cup T \cup \{S\})^*$ and *m-Pn-G* G, $x \Rightarrow y$ if and only if either $x = S$ and $[S \to y] \in P$, or $x = y_{11} X_{11} \dots y_{mn} X_{mn}$, $y = y_{11} x_{11} \dots y_{mn} x_{mn}$, where $y_{ij} \in T^*$, $X_{ij} \in N_{ij}$, $1 \leq i \leq m$, $1 \leq j \leq n$, and $[X_{i1} \to x_{i1}, \dots, X_{in} \to x_{in}] \in P$, $1 \leq i \leq m$.

If $x, y \in (N \cup T \cup \{S\})^*$ and $m \geq 0$, then $x \Rightarrow^m y$ if and only if there exists a sequence $x_0 \Rightarrow x_1 \Rightarrow \ldots \Rightarrow x_m$, $x_0 = x$, $x_m = y$. Then we say $x \Rightarrow^+ y$ if and only if there exists $m > 0$ such that $x \Rightarrow^m y$, and $x \Rightarrow^* y$ if and only if either $x = y$ or $x \Rightarrow^+ y$.

Alternatively, we define the transitive closure $\Rightarrow^+$, and the reflexive transitive closure $\Rightarrow^*$, of $\Rightarrow$ in the usual way.

The language generated by an *m-Pn-G*, G, is denoted $L(G)$ and defined as $L(G) = \{x \mid S \Rightarrow^+ x, x \in T^*\}$. A language $L \subseteq T^*$ is an *m-parallel n-right-linear simple matrix language* (*m-Pn-L*) if and only if there exists a *m-Pn-G* G such that $L = L(G)$. The family of *m-Pn-L* is denoted by $\mathcal{R}^m_{[n]}$.

3 Definitions

Let I denote the set of all positive integers and let $n \in I$.

A *n-right-linear #-rewriting system* (n-RLIN#RS) is a quadruple $H = (Q, \Sigma, s, R)$, where Q is a finite set of states, Σ is an alphabet containing # called a *bounder*, $Q \cap \Sigma = \emptyset$, $s \in Q$ is a start state, $R \subseteq Q \times I \times \{\#\} \times Q \times ((\Sigma - \{\#\})^*\# \cup (\Sigma - \{\#\})^*)$ is a finite relation whose members are called *rules*, and n denotes the number of #s in the initial configuration.

A rule $(p, i, \#, q, x) \in R$, where $i \in I$, $q, p \in Q$ and $x \in \alpha\#$ or $x \in \alpha$, where $\alpha \in (\Sigma - \{\#\})^*$, is usually written as $r{:}\, p\, {}_i\# \rightarrow q\ x$ hereafter, where r is its unique label.

A *configuration* of H is a pair from $Q \times \Sigma^*$. Let χ denote the set of all configurations of H. Let $pu\#v, quxv \in \chi$ be two configurations, $p, q \in Q$, $u, v \in \Sigma^*$, $i \in I$ and $occur(u, \#) = i - 1$. Then, H makes a *computational step* from $pu\#v$ to $quxv$ by using $r{:}\, p\, {}_i\# \rightarrow q\ x$, symbolically written $pu\#v\ {}_i\!\Rightarrow quxv\ [r]$ in H or $pu\#v \Rightarrow quxv\ [r]$ in H when position of the rewritten # symbol is not relevant or simply $pu\#v \Rightarrow quxv$ when the applied rule is irrelevant.

In the standard manner, we extend $\Rightarrow$ to $\Rightarrow^m$ and ${}_j\!\Rightarrow$ to ${}_j\!\Rightarrow^m$, for $m \geq 0$, $j > 0$; then, based on $\Rightarrow^m$ and ${}_j\!\Rightarrow^m$, we define $\Rightarrow^+$, $\Rightarrow^*$, ${}_j\!\Rightarrow^+$, and ${}_j\!\Rightarrow^*$ in the standard way. Let $\Rightarrow^m$, $\Rightarrow^+$, and $\Rightarrow^*$ denote m-step computation, non-trivial computation, and computation, respectively.

The *language generated* by the n-RLIN#RS H, $L(H)$, is defined as

$$L(H) = \{w \mid s\#^n \Rightarrow^* qw,\ q \in Q, w \in (\Sigma - \{\#\})^*\}.$$

Let k be a positive integer and σ be a initial configuration of a #-rewriting system H. H is of *index* k if for every configuration $x \in \chi$, $\sigma \Rightarrow^* qy = x$ implies $occur(y, \#) \leq k$. Notice that H of index k cannot derive a string containing more than k #s. Furthermore, notice that a k-RLIN#RS H is always of index k.

Let k be a positive integer. $\mathcal{L}(k\text{-RLIN\#RS})$ denotes the family of languages generated by k-right-linear #-rewriting systems.

A computational step is *#-erasing* if # is rewritten with a string of terminals or empty string during this step.

Let d be an n-step computation in H, for some $n \geq 0$. By d_i and ${}_td_i$, we denote the ith computational step in d and the ith computational step rewriting the tth #, respectively. t is called the *degree of step* d_i. The computation d is *successful* if d describes a computation from the initial configuration to a final configuration qw with $w \in (\Sigma - \{\#\})^*$.

Example 1. 3-RLIN#RS $H = (\{s, p, q, r, t\}, \{a, b, c, \#\}, s, R)$, where R contains

1: $s\,{}_1\# \rightarrow p\ a\#$
2: $p\,{}_2\# \rightarrow q\ b\#$
3: $q\,{}_3\# \rightarrow s\ c\#$
4: $s\,{}_1\# \rightarrow r\ a$
5: $r\,{}_1\# \rightarrow t\ b$
6: $t\,{}_1\# \rightarrow t\ c$

Obviously, $L(M) = \{a^n b^n c^n \mid n \geq 1\}$. For instance, H computes $aabbcc$ by 6-step computation d: $s\#\#\# \Rightarrow pa\#\#\#$ [1] $\Rightarrow qa\#b\#\#$ [2] $\Rightarrow sa\#b\#c\#$ [3] $\Rightarrow raab\#c\#$ [4] $\Rightarrow taabbc\#$ [5] $\Rightarrow taabbcc$ [6], where $d = {}_1d_1\,{}_2d_2\,{}_3d_3\,{}_1d_4\,{}_1d_5\,{}_1d_6$.

4 Results

We demonstrate that $\mathcal{R}^m_{[n]} \subset \mathcal{L}(mn\text{-RLIN\#RS}) = \mathcal{R}^1_{[mn]}$ and that $\mathcal{L}(n\text{-RLIN\#RS})$ with simple restriction placed on rewriting is equal to the family of right-linear languages.

Throughout this section, we only describe the construction parts of the proofs, leaving the rigorous verification of these constructions to the reader.

Lemma 1. *For every $m, n \geq 1$, $\mathcal{R}^m_{[n]} \subseteq \mathcal{L}(mn$-RLIN#RS).*

Proof. Let $G = (N_{11}, \ldots, N_{mn}, T, S, P)$ be an m-parallel n-right-linear simple matrix grammar and let $M_1, \ldots, M_m$ be mutually disjoint *matrix-rule sets*, where for every $1 \leq i \leq m$, $M_i = \{\mu\colon [X_{i1} \rightarrow \alpha_{i1}Y_{i1}, \ldots, X_{in} \rightarrow \alpha_{in}Y_{in}] \mid \mu \in P,$ $X_{ij}, Y_{ij} \in N_{ij}$, $\alpha_{ij} \in T^*$, $1 \leq j \leq n\} \cup \{\mu\colon [X_{i1} \rightarrow \alpha_{i1}, \ldots, X_{in} \rightarrow \alpha_{in}] \mid \mu \in P,$ $X_{ij} \in N_{ij}$, $\alpha_{ij} \in T^*$, $1 \leq j \leq n\}$ such that $P - \{\sigma\colon [S \rightarrow X_{11} \ldots X_{mn}] \mid \sigma \in P\} = \bigcup_{1 \leq i \leq m} M_i$.

Construction. We construct mn-right-linear #-rewriting system, $H = (Q, \Sigma,$ $s, R)$, $\Sigma = T \cup \{\#\}$, by performing following steps:

1. $Q = \{s\} \cup \{\langle \eta, \mu, l\rangle \mid \eta \in \mathit{suffixes}(X_{11} \ldots X_{mn})$, $X_{ij} \in N_{ij}$ for all $1 \leq i \leq m$, $1 \leq j \leq n$, $\mu \in M_k$, $1 \leq k \leq m$, $1 \leq l \leq n\}$, where s is a new symbol for the start state;
2. $R =$
   (i) $\{s\,{}_1\# \rightarrow \langle X_{11} \ldots X_{mn}, \mu_1, 1\rangle\ \#$
   $\mid \mu_1 \in M_1$, $X_{11} \ldots X_{mn} = \mathit{rhs}(\sigma)$, $\sigma\colon [S \rightarrow X_{11} \ldots X_{mn}] \in P\}$
   (ii) $\cup\ \{\langle Y_{11} \ldots Y_{ij-1}X_{ij} \ldots X_{mn}, \mu_i, j\rangle\ {}_{(i-1)\cdot n+j}\# \rightarrow$
   $\langle Y_{11} \ldots Y_{ij}X_{ij+1} \ldots X_{mn}, \mu_i, j+1\rangle \alpha_{ij}\#$
   $\mid \mu_i\colon [X_{i1} \rightarrow \alpha_{i1}Y_{i1}, \ldots, X_{in} \rightarrow \alpha_{in}Y_{in}] \in M_i,$
   $1 \leq i \leq m$, $1 \leq j < n\}$

(iii) $\cup \{\langle Y_{11} \dots Y_{in-1}X_{in} \dots X_{mn}, \mu_i, n\rangle \,_{i \cdot n}\# \rightarrow$
$\langle Y_{11} \dots Y_{in}X_{(i+1)1} \dots X_{mn}, \mu_{i+1}, 1\rangle \alpha_{in}\#$
$\mid 1 \leq i < m,\ \mu_{i+1} \in M_{i+1},\ \mu_i\colon [X_{i1} \rightarrow \alpha_{i1}Y_{i1}, \dots, X_{in} \rightarrow \alpha_{in}Y_{in}] \in M_i\}$

(iv) $\cup \{\langle Y_{11} \dots Y_{mn-1}X_{mn}, \mu_m, n\rangle \,_{m \cdot n}\# \rightarrow \langle Y_{11} \dots Y_{mn}, \mu_1, 1\rangle\ \alpha_{mn}\#$
$\mid \mu_1 \in M_1,\ \mu_m\colon [X_{m1} \rightarrow \alpha_{m1}Y_{m1}, \dots, X_{mn} \rightarrow \alpha_{mn}Y_{mn}] \in M_m\}$

(v) $\cup \{\langle X_{ij} \dots X_{mn}, \mu_i, j\rangle \,_1\# \rightarrow \langle X_{ij+1} \dots X_{mn}, \mu_i, j+1\rangle\ \alpha_{ij}$
$\mid \mu_i\colon [X_{i1} \rightarrow \alpha_{i1}, \dots, X_{in} \rightarrow \alpha_{in}] \in M_i,\ 1 \leq i \leq m,\ 1 \leq j < n\}$

(vi) $\cup \{\langle X_{in} \dots X_{mn}, \mu_i, n\rangle \,_1\# \rightarrow \langle X_{(i+1)1} \dots X_{mn}, \mu_{i+1}, 1\rangle\ \alpha_{in}$
$\mid 1 \leq i < m,\ \mu_{i+1} \in M_{i+1},\ \mu_i\colon [X_{i1} \rightarrow \alpha_{i1}, \dots, X_{in} \rightarrow \alpha_{in}] \in M_i\}$

(vii) $\cup \{\langle X_{mn}, \mu_m, n\rangle \,_1\# \rightarrow \langle \varepsilon, \mu_m, n\rangle\ \alpha_{mn}$
$\mid \mu_m\colon [X_{m1} \rightarrow \alpha_{m1}, \dots, X_{mn} \rightarrow \alpha_{mn}] \in M_m\}$.

Basic Idea. H simulates each derivation step in G using the states to hold necessary information about each step. Instead of parallelism, the rules from G are divided into the sets of matrices, M_i. Each state from Q contains a string of nonterminals, a matrix label μ_i, and a rule-index indicating next rule to be applied. The last rule in M_i changes the state of H so a matrix from M_{i+1} can be used. Tthe last rule of a matrix from M_m changes the state of H so it can apply the first rule of a matrix from the very first M_1.

The rules from P of the form $X_{ij} \rightarrow \alpha_{ij}Y_{ij}$ change nonterminals and the rules of the form $X_{ij} \rightarrow \alpha_{ij}$ remove those nonterminals in the string stored in the first component of the state. When there are no nonterminals left, the system can make no more steps and the computation ends. □

Lemma 2. *For every $n \geq 1$, $\mathcal{L}(n\text{-}RLIN\#RS) \subseteq \mathcal{R}^1_{[n]}$.*

Proof. Let $H = (Q, \Sigma, s, R)$ be an n-right-linear #-rewriting system.
Construction. We construct an m-parallel n-right-linear simple matrix grammar $G = (N_{11}, \dots, N_{mn}, T, S, P)$ with $m = 1$ by performing the following steps:

1. $T = \Sigma - \{\#\}$
2. $N_{1i} = \{\langle i, j, q\rangle \mid q \in Q,\ 1 \leq j \leq i\} \cup \{X_i\}$ for every $1 \leq i \leq n$, where X_i is a new nonterminal.
3. Add $S \rightarrow \langle 1, 1, s\rangle\langle 2, 2, s\rangle \dots \langle n, n, s\rangle$ to P.
4. For every rule $r\colon p\,_j\# \rightarrow q\ \alpha\# \in R$ add $[\eta_1, \dots, \eta_{i-1}, \langle i, j, p\rangle \rightarrow \alpha\langle i, j, q\rangle, \eta_{i+1}, \dots, \eta_n]$ into P, where for every $k \in \{1, \dots, n\} - \{i\}$ and $1 \leq k' \leq k$, η_k is of the form $\langle k, k', p\rangle \rightarrow \langle k, k', q\rangle$ or $X_k \rightarrow X_k$.
5. For every rule $r\colon p\,_j\# \rightarrow q\ \alpha \in R$ add $[\eta_1, \dots, \eta_{i-1}, \langle i, j, p\rangle \rightarrow \alpha X_i, \eta_{i+1}, \dots, \eta_n]$ into P, where
for every $1 \leq k < i$ and $1 \leq k' \leq k$, η_k is of form $\langle k, k', p\rangle \rightarrow \langle k, k', q\rangle$ or $X_k \rightarrow X_k$ and
for every $i < l \leq n$ and $1 \leq l' \leq n$, η_l is of form $\langle l, l', p\rangle \rightarrow \langle l, l'-1, q\rangle$ or $X_l \rightarrow X_l$.
6. Add $[X_1 \rightarrow \varepsilon, X_2 \rightarrow \varepsilon, \dots, X_n \rightarrow \varepsilon]$ to P.

Basic Idea. G simulates each computational step in H as follows. Every nonterminal has three components. To make the nonterminal alphabets N_{1i}, ..., N_{1n} disjoint, the first component contains the nonterminal-alphabet-index. The second component represents the position of the corresponding bounder in the H's current configuration. The third component consists of information about the states of H.

In addition, the auxiliary nonterminals X_1, ..., X_n that do not hold any information about states or #s are introduced. They allow us to have all matrices of the same size n as required by the definition of m-Pn-G.

R's rules of the form $p\,{}_j\# \to q\ \alpha\#$ change the state-related information inside of all nonterminals except for those of form X_i. R's rules of the form $p\,{}_j\# \to q\ \alpha$ do the same job appart from rewriting a nonterminal $\langle i, j, p\rangle$ into X_i and reindexing nonterminals following the rewritten one. This simulates removing of a #.

When all nonterminals are of the form X_i, the rule $[X_1 \to \varepsilon,\ X_2 \to \varepsilon,\ \ldots,\ X_n \to \varepsilon]$ removes all nonterminals from the sentential form. □

Theorem 1. *For every $m, n \geq 1$ such that $m{+}n > 1$, $\mathcal{R}^m_{[n]} \subset \mathcal{L}(mn\text{-}RLIN\#RS) = \mathcal{R}^1_{[mn]}$.*

Proof. Recall that $\mathcal{R}^{mn}_{[1]} \subset \mathcal{R}^m_{[n]} \subset \mathcal{R}^1_{[mn]}$, for every $m + n > 1$ (see Theorem 10 in [5]). Thus, Theorem 1 follows from $\mathcal{R}^m_{[n]} \subset \mathcal{R}^1_{[mn]}$, Lemmas 1 and 2. ■

Corollary 1. *For every $n \geq 1$, $\mathcal{L}(n\text{-}RLIN\#RS) \subset \mathcal{L}(n + 1\text{-}RLIN\#RS)$.*

Proof. Recall that $\mathcal{R}^m_{[n]} \subset \mathcal{R}^m_{[n+1]}$, for every $m, n \geq 1$ (see Theorem 8 in [5]). Since Theorem 1 proves $\mathcal{L}(n\text{-RLIN\#RS}) = \mathcal{R}^1_{[n]}$, the corollary holds. ■

Before we present Theorem 2, we give an insight into the implication it contains to make it easier to understand. To illustrate the denotation of a computational step by ${}_ud_i$, we write $\overset{{}_ud_i}{\Rightarrow}$. The implication restricts every successful computation in a #-rewriting system. Let d: $s\#^n = p_0w_0 \overset{{}_{u_1}d_1}{\Rightarrow} p_1w_1 \overset{{}_{u_2}d_2}{\Rightarrow} \ldots \overset{{}_{u_i}d_i}{\Rightarrow} p_iw_i \overset{{}_{u_{i+1}}d_{i+1}}{\Rightarrow} \ldots \overset{{}_{u_j}d_j}{\Rightarrow} p_jw_j \overset{{}_{u_{j+1}}d_{j+1}}{\Rightarrow} \ldots \overset{{}_{u_{|d|}}d_{|d|}}{\Rightarrow} p_{|d|}w_{|d|}$ be a successful computation, where $1 \leq i \leq j \leq |d|$, $u = u_i$, $v = u_j$, and $w_d \in (\Sigma - \{\#\})^*$.

If $u = v$ in ${}_ud_i$ and ${}_vd_j$ then there are allowed only two cases:

(a) all computational steps between ${}_ud_i$ and ${}_vd_j$, denoted by ${}_zd_k$ for all $i \leq k \leq j$, rewrite just zth bounder and nothing else, so $z = u = v$;
(b) there can be only one exception in (a) such that ${}_ld_h$, $i < h < j$, is #-erasing computational step.

Theorem 2. *Let every successful computation d in an n-right-linear #-rewriting system H, $n \geq 1$, satisfy this implication: if $1 \leq i \leq j \leq |d|$ and $u = v$ in ${}_ud_i$ and ${}_vd_j$, then either $z = u$ in ${}_zd_k$ for all $i \leq k \leq j$ or d_h is #-erasing for some $h \in \{i + 1, \ldots, j - 1\}$. Then, $L(H)$ is right-linear.*

Proof. Let $H = (Q, \Sigma, s, R)$ be a n-right-linear $\#$-rewriting system satisfying the preceding implication, for some $n \geq 1$.

Construction. We transform H to an equivalent right-linear grammar $G = (V, T, P, S)$ by performing the following procedure:

For every $1 \leq i \leq n$, $p, q \in Q$, construct auxiliary sets ${}^p_qR_i = \{r \mid r \in alph(\rho),\ \rho \in R^*,\ p\gamma\ {}_i{\Rightarrow}^*\ q\delta\ [\rho],\ occur(\gamma, \#) = occur(\delta, \#)\}$ and ${}^p_q\bar{R}_i = \{p\ {}_i\# \rightarrow q\ \alpha \in R \mid \alpha \in (\Sigma - \{\#\})^*\}$. Then, $\mathcal{Z} = \bigcup_{i \geq 1, p, q \in Q} \{{}^p_qR_i, {}^p_q\bar{R}_i\}$.

1. $T = \Sigma - \{\#\}$,
2. $V = N \cup T \cup \{S\}$, where S is a new symbol and N contains nonterminals introduced by the following construction of P,
3. $P = \bigcup_{1 \leq l \leq 5} P_l$, where sets P_1 through P_5 are constructed in the following way:
   (i) initialization: $P_1 = \{S \rightarrow \langle \#^n, i, s\rangle \mid 1 \leq i \leq n\}$;
   (ii) preparation: $P_2 = \{\langle \nabla_1\eta_1\nabla\eta_2 \ldots \nabla_i\eta_i\nabla_{i+1}\eta_{i+1} \ldots \nabla_n\eta_n, i, p\rangle \rightarrow$
   $\langle \nabla_1\eta_1\nabla\eta_2 \ldots \nabla_i\eta_i{}^p_qR_{i'}\nabla_{i+1}\eta_{i+1} \ldots \nabla_n\eta_n, j, q\rangle$
   $\mid {}^p_qR_{i'} \neq \emptyset$, $1 \leq j \leq n$, $\eta_t \in \mathcal{Z}^*$, $\nabla_t \in \{\#, \bar{\#}\}$ for $1 \leq t \leq n$, $\nabla_i \neq \bar{\#}$,
   $i' = occur(\nabla_1\eta_1 \ldots \nabla_i, \#)\}$
   $\cup\ \{\langle \nabla_1\eta_1 \ldots \nabla_{i-1}\eta_{i-1}\nabla_i\eta_i\nabla_{i+1} \ldots \nabla_n\eta_n, i, p\rangle \rightarrow$
   $\langle \nabla_1\eta_1 \ldots \nabla_{i-1}\eta_{i-1}\bar{\#}\eta_i{}^p_q\bar{R}_{i'}\nabla_{i+1}\eta_{i+1} \ldots \nabla_n\eta_n, j, q\rangle$
   $\mid {}^p_q\bar{R}_{i'} \neq \emptyset$, $1 \leq j \leq n$, $\eta_t \in \mathcal{Z}^*$, $\nabla_t \in \{\#, \bar{\#}\}$ for $1 \leq t \leq n$, $\nabla_i \neq \bar{\#}$,
   $i' = occur(\nabla_1\eta_1 \ldots \nabla_i, \#)\}$;
   (iii) latch:
   $P_3 = \{\langle \gamma, i, p\rangle \rightarrow \langle \gamma, q\rangle \mid p, q \in Q,\ \# \notin alph(\gamma),\ A \rightarrow \langle \gamma, i, p\rangle \in P_2\}$;
   (iv) simulation of G's derivation step ($\eta_t \in \mathcal{Z}^*$ for every $1 \leq t \leq n$):
   $P_4 = \{\langle \bar{\#}{}^p_qR_{i'}\eta_i \ldots \bar{\#}\eta_n, p'\rangle \rightarrow \alpha\langle \bar{\#}{}^p_qR_{i'}\eta_i \ldots \bar{\#}\eta_n, q'\rangle$
   $\mid p'\ {}_{i'}\# \rightarrow q'\ \alpha\# \in {}^p_qR_{i'}$, $\alpha \in (\Sigma - \{\#\})^*$, $1 \leq i \leq n$, ${}^p_qR_{i'} \in \mathcal{Z}\}$
   $\cup\ \{\langle \bar{\#}{}^p_qR_{i'}\eta_i \ldots \bar{\#}\eta_n, p'\rangle \rightarrow \alpha\langle \bar{\#}\eta_i \ldots \bar{\#}\eta_n, q\rangle$
   $\mid p'\ {}_{i'}\# \rightarrow q\ \alpha\# \in {}^p_qR_{i'}$, $\alpha \in (\Sigma - \{\#\})^*$, $1 \leq i \leq n$, ${}^p_qR_{i'} \in \mathcal{Z}\}$
   $\cup\ \{\langle \bar{\#}{}^p_q\bar{R}_{i'}\bar{\#}\eta_{i+1} \ldots \bar{\#}\eta_n, p\rangle \rightarrow \alpha\langle \bar{\#}\eta_{i+1}\bar{\#}\eta_n, q'\rangle$
   $\mid p\ {}_{i'}\# \rightarrow q\ \alpha \in {}^p_q\bar{R}_{i'}$, $\alpha \in (\Sigma - \{\#\})^*$, $1 \leq i \leq n$, ${}^p_q\bar{R}_{i'} \in \mathcal{Z}\}$;
   (v) finalization: $P_5 = \{\langle \varepsilon, p\rangle \rightarrow \varepsilon \mid p \in Q\}$.

The conversion of a right-linear grammar, G, to an n-right-linear $\#$-rewriting system, H, is simple and left to the reader.

Basic Idea. Every ${}^p_qR_i$ represents a set of rules which can make a computation of degree i leading from state p to q in H. In every sentential from in G, there is only one occurrence of a nonterminal which is composed of three components:

(1) γ—the finite prescription string for the driven simulation, $\gamma \in (\#\mathcal{Z}^*)^+$;
(2) i—the position of the occurrence of active $\#$ in H's current configuration;
(3) p—the currently simulated state of H.

There are non-deterministically generated prescription substrings behind every corresponding bounder in H's configuration in the preparation phase. These substrings η_t are of the form $\mathcal{Z}^*$.

In the third step, the nonterminal's second component is removed in G to ensure to ensure that the rules of P_2 cannot be used anymore.

By rules constructed in the fourth step, the generation of terminals is done with correspondence to the γ-prescription string. Each completed ${}_q^pR_i$ is removed from γ until γ is the empty string. Then, the only nonterminal in the sentential form of G is rewritten to the empty string by a rule from P_5 and a string of terminals is reached. ■

5 Conclusion

The present paper has discussed simple language-defining devices that represent a combination of both automata and grammars. These devices characterize some well-known infinite hierarchies of formal language families in a very natural way. Consequently, they are obviously closely related to some classical results about formal languages, on which they shed light in an alternative way. Therefore, this paper suggests their further investigation in the future. Specifically, this investigation should pay a special attention to the following open problem areas:

Determinism. This paper has discussed a general version of n-right-linear #-rewriting systems, which work non-deterministically. Undoubtedly, the future investigation of these systems should study their deterministic versions, which can make no more than one computational step from any configuration because these deterministic versions are crucial in practice.

Infinite Index. Consider #-rewriting systems that are not of finite index. What is the language family defined by them.

Acknowledgement. This work was supported by the Czech Ministry of Education, Youth and Sports grant MŠMT 2C06008 *Virtual Laboratory of Microprocessor Technology Application*, MŠMT FRVŠ grant 673/2007/G1, and the Czech Ministry of Education under the Research Plan No. MSM 0021630528.

References

1. J. Dassow, G. Păun, *Regulated Rewriting in Formal Language Theory.* Springer, New York, 1989, 308 p., ISBN 0-38751-414-7.
2. Z. Křivka, A. Meduna, Generalized #-Rewriting Systems of Finite Index. In: *Information Systems and Formal Models (Proceedings of 2nd International Workshop on Formal Models (WFM'07))*, 2007, pp. 197-204, ISBN 978-807248-006-7.
3. Z. Křivka, A. Meduna, R. Schönecker, Generation of Languages by Rewriting Systems that Resemble Automata. In: *International Journal of Foundations of Computer Science* Vol. 17, No. 5, 2006, pp. 1223-1229.
4. A. Meduna, *Automata and Languages: Theory and Applications.* Springer, London, 2000, 916 p., ISBN 1-85233-074-0.
5. D. Wood, m-Parallel n-Right Linear Simple Matrix Languages. In: *Utilitas Mathematica* Vol. 8, 1975, pp. 3-28.
6. G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages: Linear Modeling*, Volume 2. Springer, Berlin, 1997, 873 p., ISBN 3-540-60420-0.