Generalized #-Rewriting Systems of Finite Index

Zbyněk Křivka and Alexander Meduna

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

Abstract. This paper discusses a generalized version of #-rewriting systems with context rules. It demonstrates that this context-based generalization does not affect the generative power of #-rewriting systems of finite index. A new characterization of the infinite hierarchy of language families generated by programmed grammars of finite index is obtained.

Keywords: #-rewriting systems, context-based generalization, infinite hierarchy, finite index.

1 Introduction

In the formal language theory, there exist language-defining devices having features of both grammars and automata (see [1], [3], and [6]). Recently, this theory has introduced another device of this kind—*#-rewriting systems* (see [4]). Indeed, on the one hand, like grammars, they are generative devices. On the other hand, like automata, they use finitely many states. Recall that these systems of finite index characterize the well-known infinite hierarchy of language families resulting from programmed grammars of finite index (see Theorems 3.1.2i and 3.1.7 in [2]).

The original version of #-rewriting systems is based upon rules of the form $p\ {}_i\# \rightarrow q\ \gamma$, where p, q are states, i is a positive integer, and γ is a non-empty string. By using this rule, the system rewrites ith # with γ and, simultaneously, changes the current state p to q. In the present paper, we discuss a generalized version of #-rewriting systems that uses rules of the form $p\ {}_i\alpha\#\beta \rightarrow q\ \alpha\gamma\beta$, where α and β are strings and the other symbols have the same meaning as above. This generalized rule is applicable to # if this # occurs in the α-β context; in other aspects, the application is analogical to the original version.

As its main result, this paper demonstrates that the generalization under discussion does not affect the generative power of #-rewriting systems of finite index, so we obtain an alternative characterization of the infinite hierarchy of language families generated by programmed grammars of finite index (see [4], and Theorems 3.1.2i and 3.1.7 in [2]).

This result is of some interest when compared, for instance, to a similar generalization in terms of the classical Chomsky Hierarchy, in which grammars with the generalized rules are much stronger than ordinary context-free grammars.

On the other hand, notice that the language family generated by #-rewriting systems of finite index is incomparable with the family of context-free languages. More specifically, a context-free Dyck language is not generated by any #-rewriting systems of finite index and vice versa Example 1 shows that the family of languages generated by #-rewriting systems of finite index contains at least one non-context-free language.

2 Preliminaries

This paper assumes that the reader is familiar with the formal language theory (see [5], [7]). 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 every $i \geq 0$, $suffix(w, i)$ is w's suffix of length i if $|w| \geq i$, and $suffix(w, i) = w$ if $i \geq |w| + 1$. Let $suffixes(w) = \{suffix(w, j) \mid 0 \leq j \leq |w|\}$. For every $i \geq 0$, $prefix(w, i)$ is w's prefix of length i if $|w| \geq i$, and $prefix(w, i) = w$ if $i \geq |w| + 1$. Let $prefixes(w) = \{prefix(w, j) \mid 0 \leq j \leq |w|\}$.

Let Ψ denote the set of all non-negative integers and let $m \in \Psi$. Set $I = \Psi - \{0\}$. Let $K \subseteq \Psi$ be a finite set. Define $\max(K)$ as the smallest integer k such that for all $h \in K$, $k \geq h$. Define $\min(K)$ as the greatest integer k such that for all $h \in K$, $k \leq h$.

For a subset S of partially ordered set $(\Psi, \leq)$, a *supremum* of S is an element $p \in \Psi$ such that $x \leq p$ for all $x \in S$, and for any $q \in \Psi$ such that $x \leq q$ for all $x \in S$ it holds that $p \leq q$.

The *infimum* of the subset S is an element $p \in \Psi$ such that $p \leq x$ for all $x \in S$, and for any $q \in \Psi$ such that $q \leq x$ for all $x \in S$ it holds that $q \leq p$.

If the supremum of a set S exists, it is denoted as $\sup(S)$; infimum is denoted by $\inf(S)$.

A *context-free 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 $A \in (V - T)$, $v \in V^*$ and q is a 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 $q_1, q_2, \ldots, q_m$, we symbolically write $x \Rightarrow^m y\ [q_1 q_2 \ldots q_m]$. The *language of* G, $L(G)$, is defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. A language, L, is *context-free* if and only if $L = L(G)$, where G is a context-free grammar.

A *programmed grammar* (see page 28 in [2]) 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 $q\colon A \to v, g(q)$, where $q\colon A \to v$ is a context free rule labeled by q and $g(q)$ is a set of rule labels

associated with this rule. After a derivation step, symbolically denoted by $\Rightarrow$, according a rule of this form in an ordinary context-free way, in the next step a rule labeled by a label from $g(q)$ has to be applied. In the standard manner, we define $\Rightarrow^m$, where $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language of G, $L(G)$, is defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$.

Let G be a grammar, and let T and S be its terminal alphabet and start symbol, respectively. For a derivation $D\colon S = w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_r = w \in T^*$, $r \geq 1$, according to G, we set $Ind(D, G) = \max(\{occur(w_i, V - T) \mid 1 \leq i \leq r\})$, and for $w \in T^*$, we define $Ind(w, G) = \min(\{Ind(D, G) \mid D$ is a derivation for w in $G\})$. The *index of grammar* G (see page 151 in [2]) is defined as $Ind(G) = \sup\{Ind(w, G) \mid w \in L(G)\}$. For a language L in the family $\mathcal{L}(X)$ of languages generated by grammars of some type X, we define $Ind_X(L) = \inf\{Ind(G) \mid L(G) = L, G$ is of type $X\}$. For a family $\mathcal{L}(X)$, we set $\mathcal{L}_n(X) = \{L \mid L \in \mathcal{L}(X)$ and $Ind_X(L) \leq n\}$, $n \geq 1$.

3 Definitions

A *generalized #-rewriting system* (G#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 and $R \subseteq Q \times I \times \Sigma^+ \times Q \times \Sigma^*$ is a finite relation whose members are called *rules*. A rule is usually written as $r\colon p\ {}_i\alpha\#\beta \rightarrow q\ \alpha\gamma\beta \in R$ hereafter, where r is its unique label, $i \in I$, $q, p \in Q$, and $\alpha, \beta, \gamma \in \Sigma^*$.

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

In the standard manner, we extend $\Rightarrow$ to $\Rightarrow^m$ and ${}_j\Rightarrow$ to ${}_j\Rightarrow^m$, for $m \geq 0$; then, based on $\Rightarrow^m$ and ${}_j\Rightarrow^m$, we define $\Rightarrow^+$, $\Rightarrow^*$, ${}_j\Rightarrow^+$, and ${}_j\Rightarrow^*$ in the standard way.

The *language generated* by G#RS H, $L(H)$, is defined as

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

As special case of G#RS, if every $r\colon p\ {}_i\alpha\#\beta \rightarrow q\ \alpha\gamma\beta \in R$ satisfies that $\alpha = \beta = \varepsilon$ then H is called *context-free #-rewriting system* (CF#RS).

Let k be a positive integer. H is of *index* k if for every configuration $x \in \chi$, $s\# \Rightarrow^* qy = x$ implies $occur(y, \#) \leq k$. Notice that H of index k cannot derive a string containing more than k #s.

For G#RS H, $\max_L(H)$ and $\max_R(H)$ denote the maximum length of left-hand and right-hand side of rules, respectively. Precisely, let $H = (Q, \Sigma, s, R)$

be a G#RS, $\max_L(H) = \max(\{|\alpha| \mid p\ {}_i\alpha \rightarrow q\ \beta \in R\})$ and $\max_R(H) = \max(\{|\beta| \mid p\ {}_i\alpha \rightarrow q\ \beta \in R\})$.

Let k be a positive integer. $\mathcal{L}_k(G\#RS)$, $\mathcal{L}_k(CF\#RS)$, and $\mathcal{L}_k(P, CF)$ denote the families of languages generated by generalized #-rewriting systems of index k, context-free #-rewriting systems of index k, and derived by programmed grammars of index k, respectively. $\mathcal{L}(G\#RS)$ and $\mathcal{L}(CF\#RS)$ denote the family of languages generated by generalized #-rewriting systems and context-free #-rewriting systems, respectively.

Example 1. CF#RS $H_1 = (\{s, p, q, f\}, \{a, b, c, \#\}, s, R_1)$, where R_1 contains

1: $s\ {}_1\# \rightarrow p\ \#\#$
2: $p\ {}_1\# \rightarrow q\ a\#b$
3: $q\ {}_2\# \rightarrow p\ c\#$
4: $p\ {}_1\# \rightarrow f\ ab$
5: $f\ {}_1\# \rightarrow f\ c$

For instance, H_1 computes $aabbcc$ as $s\# \Rightarrow p\#\#$ [1] $\Rightarrow qa\#b\#$ [2] $\Rightarrow pa\#bc\#$ [3] $\Rightarrow faabbc\#$ [4] $\Rightarrow faabbcc$ [5].

Example 2. G#RS $H_2 = (\{s, p, q\}, \{a, b, c, \#\}, s, R_2)$, where R_2 contains

1: $s\ {}_1\# \rightarrow s\ a\#\#$
2: $s\ {}_2a\#\# \rightarrow p\ a\#b\#c$
3: $p\ {}_1a\# \rightarrow q\ aa\#$
4: $q\ {}_2b\#c \rightarrow p\ bb\#cc$
5: $p\ {}_1a\# \rightarrow p\ a$
6: $p\ {}_1b\#c \rightarrow p\ bc$

For instance, H_2 computes $aabbcc$ as $s\# \Rightarrow sa\#\#$ [1] $\Rightarrow pa\#b\#c$ [2] $\Rightarrow qaa\#b\#c$ [3] $\Rightarrow paa\#bb\#cc$ [4] $\Rightarrow paabb\#cc$ [5] $\Rightarrow paabbcc$ [6].

Obviously, H_1 and H_2 are of index 2. Both systems describe the same language $L(H_1) = L(H_2) = \{a^n b^n c^n \mid n \geq 1\}$ which belongs into $\mathcal{L}(CF\#RS) - \mathcal{L}(CF)$.

4 Results

The first part of the section establishes an infinite hierarchy of language families resulting from the generalized #-rewriting systems defined in the previous section.

Throughout this section, we only describe the construction part of the proof, leaving the complete version of this proof to the reader.

Lemma 1. *For every $k \geq 1$, $\mathcal{L}_k(CF\#RS) = \mathcal{L}_k(G\#RS)$.*

Proof.
We only need to prove that $\mathcal{L}_k(G\#RS) \subseteq \mathcal{L}_k(CF\#RS)$.

Construction. Let $k \geq 1$ be a positive integer. Let $H = (Q, T \cup \{\#\}, s, R)$ be a generalized #-rewriting system of index k, where $\Sigma = T \cup \{\#\}$, $\# \notin T$. Let $\mu = \max(\{\max_L(H), \max_R(H)\})$ and let \$ be a new symbol, $\$ \notin Q \cup \Sigma$. We construct the context-free #-rewriting system of index k, $H' = (Q', T \cup \{\#\}, s', R')$, where components Q', s', and R' are constructed as follows:

1. $s' := \langle s, \# \rangle$;
2. $Q' := \{\langle p, y_0 \# y_1 \# \dots y_{h-1} \# y_h \rangle$
   $\mid p \in Q,\ 1 \leq h \leq k,$
   $y_j = y'_j \nabla_j y''_j,\ y'_j, y''_j \in T^*,\ \nabla_j \in \{\varepsilon, \$\},$
   $|y'_j| \leq 2\mu,\ |y''_j| \leq 2\mu,\ 0 \leq j \leq h\}$;
3. $R' := \{\langle p, x_0 \# \dots \# x'_j \nabla_j x''_j \# \dots \# x_h \rangle\ {}_1\# \to \langle p, x_0 \# \dots \# y'_j \$ y''_j \# \dots \# x_h \rangle\ \#$
   $\mid \langle p, x_0 \# \dots \# x'_j \nabla_j x''_j \# \dots \# x_h \rangle, \langle p, x_0 \# \dots \# y'_j \$ y''_j \# \dots \# x_h \rangle \in Q',$
   $\nabla_j \in \{\varepsilon, \$\},\ y'_j \in \mathit{prefixes}(x'_j),\ y''_j \in \mathit{suffixes}(x''_j),$
   $x_j \in T^*,\ 1 \leq h \leq k,\ 0 \leq j \leq h,$ where $x'_0 = y'_0 = x''_h = y''_h = \varepsilon\}$;
4. For every rule $r\colon p\ {}_i\alpha\#\beta \to q\ \alpha\gamma\beta \in R$, add the following set to R':
   $\{\langle p, \eta'\alpha\#\beta\eta''\rangle\ {}_i\# \to \langle q, \eta'\alpha\gamma\beta\eta''\rangle\ \gamma$
   $\mid \mathit{occur}(\eta'\alpha, \#) = i - 1,$
   $\langle p, \eta'\alpha\#\beta\eta''\rangle, \langle q, \eta'\alpha\gamma\beta\eta''\rangle \in Q'\ \}$.

Basic Idea. By several computational steps, H' simulates a single step in H. Inside of every state of the form $\langle p, \eta \rangle$ occurring in a configuration of H', it is recorded

(1) p—the current state of H;
(2) η—the context string of H's current configuration that represents context of each # of length at most μ on both sides of #.

The simulation is done in two steps by rules introduced in the third and forth step of the above construction as follows.

(a) Each substring of terminals between two #s can be non-deterministically shortened by the rules constructed in 3 and the position of the shortening is marked by \$ to reflect this in the subsequent context checks. This shortening is necessary only to make enough space in the context string for the rule application in the next step.
(b) By each of H's generalized rules is rewritten only one # with respect to its surrounding left and right context. In this step, the rules of the context-free form constructed in 4 are applied and, thereby, simulate H's behavior thanks to context checks, which are managed by the second component of states in H'.

The simulation is completed when the string of terminals is obtained.

Let h be a homomorphism from a configuration of context-free #-rewriting system to corresponding configuration of generalized #-rewriting system defined as $h(\langle p, \eta\rangle z) = pz$.

Claim 1. If $s\# \Rightarrow^m pz_0\#z_1\#\dots\#z_h$ in H, then $\langle s, \#\rangle\# \Rightarrow^r \langle p, x_0\#x_1\#\dots\#x_h\rangle\ z_0\#z_1\dots\#z_h\ [r'_1r'_2\dots r'_r]$ in H' where $x_i = x'_i\nabla_i x''_i$, $\nabla_i \in \{\varepsilon, \$\}$, $x'_i \in \mathit{prefixes}(z_i)$, $x''_i \in \mathit{suffixes}(z_i)$, $|x_i| \leq 4\mu + 1$, $x_i, z_i \in T^*$, $0 \leq i \leq h$, $1 \leq h \leq k$, for $m \geq 0$.

Proof. This claim is established by induction on m.

Basis: Let $m = 0$. For $s\# \Rightarrow^0 s\#$ in H there exists $s'\# \Rightarrow^0 s'\#$ in H', where $s' = \langle s, \#\rangle$.

Induction Hypothesis: Suppose that Claim 1 holds for all computational steps of length m or less for some $m \geq 0$.

Induction Step: Consider $s\# \Rightarrow^{m+1} qz'$, where $z' \in \Sigma^*$. Express $s\# \Rightarrow^{m+1} qz'$ as $s\# \Rightarrow^m pz\ [r_1r_2\dots r_m]$, where $z = z_0\#z_1\#\dots\alpha\#\beta\dots\#z_h$, $\#$ between α and β is the ith bounder and $r_1, \dots, r_m, r_{m+1} \in \mathit{lab}(R)$, and $pz \Rightarrow qz'\ [r_{m+1}]$. For $r_{m+1}\colon p\,{}_i\alpha\#\beta \to q\ \alpha\gamma\beta$ is z' of the form: $z' = z_0\#z_1\#\dots\alpha\gamma\beta\dots\#z_h$, for $z_0, \dots, z_h \in T^*$.

Based on the induction hypothesis, there exists $\langle s, \#\rangle\# \Rightarrow^r \langle p, x_0\#x_1\#\dots\alpha\#\beta\dots\#x_h\rangle\ z_0\#\dots\alpha\#\beta\dots z_{h-1}\#z_h\ [r'_1r'_2\dots r'_r] \Rightarrow^* \langle p, y_0\#y_1\#\dots\alpha\#\beta\dots\#y_h\rangle\ z_0\#\dots\alpha\#\beta\dots z_{h-1}\#z_h\ [\rho] \Rightarrow \langle q, y_0\#y_1\#\dots\alpha\gamma\beta\dots\#y_h\rangle\ z_0\#\dots\alpha\gamma\beta\dots z_{h-1}\#z_h\ [r'_{r+1}]$, where ρ is a sequence of rules constructed in step 3, $|\rho| \geq 0$, $r \geq 1$, $r'_j \in \mathit{lab}(R')$, $1 \leq j \leq r+1$, $x_j, y_j \in T^*\{\varepsilon, \$\}T^*$, $z_j \in T^*$, $0 \leq j \leq h$, $\mathit{occur}(y_0\#y_1\#\dots\alpha, \#) = \mathit{occur}(z_0\#z_1\#\dots\alpha, \#) = i-1$, $\alpha = \bar{x}_{i-\bar{\alpha}}\#x_{i-\bar{\alpha}+1}\dots x_{i-2}\#x_{i-1}$, $\bar{\alpha} = \mathit{occur}(\alpha, \#) + 1$, $\bar{x}_{i-\bar{\alpha}} \in \mathit{suffixes}(x_{i-\bar{\alpha}})$, $\beta = x_i\#x_{i+1}\dots x_{i+\bar{\beta}-1}\#\bar{x}_{i+\bar{\beta}}$, $\bar{\beta} = \mathit{occur}(\beta, \#)$, $\bar{x}_{i+\bar{\beta}} \in \mathit{prefixes}(x_{i+\bar{\beta}})$, and $r'_{r+1}\colon \langle p, y_0\#y_1\#\dots\alpha\#\beta\dots\#y_h\rangle\,\#\to\langle q, y_0\#y_1\#\dots\alpha\gamma\beta\dots\#x_h\rangle\ \gamma \in R'$ is introduced by step 4.

Therefore, $h(\langle p, y_0\#y_1\#\dots\alpha\#\beta\dots\#y_h\rangle\ z_0\#\dots\alpha\#\beta\dots z_{h-1}\#z_h) = pz$ and $h(\langle q, y_0\#y_1\#\dots\alpha\gamma\beta\dots\#y_h\rangle\ z_0\#\dots\alpha\gamma\beta\dots z_{h-1}\#z_h) = qz'$, so the claim holds.

Claim 2. If $s\# \Rightarrow^z pw$ in H, then $\langle s, \#\rangle\# \Rightarrow^* \langle p, \eta\rangle w$ in H for some $z \geq 0$, $w \in T^*$.

Proof. Consider Claim 1 for $h = 0$. At this point, if $s\# \Rightarrow^z pz_0$, then $\langle s, \#\rangle\# \Rightarrow^* \langle p, x_0\rangle z_0$ and so $z_0 = w$.

□

Theorem 1. *For every $k \geq 1$, $\mathcal{L}_k(G\#RS) \subset \mathcal{L}_{k+1}(G\#RS)$.*

Proof.
$\mathcal{L}_k(G\#RS) = \mathcal{L}_k(CF\#RS)$ follows from Lemma 1. Recall that $\mathcal{L}_k(P, CF) = \mathcal{L}_k(CF\#RS)$ (see [4]) and $\mathcal{L}_k(P, CF) \subset \mathcal{L}_{k+1}(P, CF)$ for every $k \geq 1$ (see Theorems 3.1.2i and 3.1.7 in [2]). Thus, Theorem 1 holds. □

5 Conclusion

The present paper has discussed generalized language-defining device that represents a combination of both automata and grammars. This device characterizes well-known infinite hierarchy of formal language families in a very natural way. Consequently, it is obviously closely related to some classical results about formal languages, on which it sheds light in an alternative way. Therefore, this paper suggests its further investigation in the future. Specifically, this investigation should pay a special attention to the following open problem areas:

Unrestricted Rules. Observe that Theorem 1 does not hold if the rules are of the form $p\ {}_i\alpha \rightarrow q\ \beta$, where $p, q \in Q$, $i \in I$, $\alpha, \beta \in \Sigma^*$, $occur(\alpha, \#) \geq 1$ and computation step is defined as follows:

Let $pu\alpha'\#\alpha''v$, $qu\beta v$ be two configurations, $p, q \in Q$, $u, v \in \Sigma^*$, $i \in I$ and $occur(u\alpha', \#) = i - 1$. Then, #-rewriting system with unrestricted rules makes a *computational step* from $pu\alpha'\#\alpha''v$ to $qu\beta v$ by using the unrestricted rule $r\colon p\ {}_i\alpha'\#\alpha'' \rightarrow q\ \beta$.

Let us demonstrate this observation on the following example.

Example 3. In [2], Theorem 3.1.7 proves that $L_k \in \mathcal{L}_k(CF\#RS)$ and $L_k \notin \mathcal{L}_{k-1}(CF\#RS)$, where $L_k = \{b(a^ib)^{2k} \mid i \geq 1\}$.

Consider $G\#RS$ with unrestricted rules $H = (\{s, s', p, q, r, r', f\}, \{a, b, \#\}, s, R)$, where R contains

1: $s\ {}_1\# \rightarrow s'\ \#\#a\#\#$
2: $s'\ {}_2\# \rightarrow s'\ \#a$
3: $s'\ {}_1\# \rightarrow p\ \#$
4: $p\ {}_2\#a \rightarrow q\ a\#$
5: $q\ {}_3\# \rightarrow p\ \#a$
6: $p\ {}_2\#\# \rightarrow r\ \#\#$
7: $r\ {}_1\# \rightarrow r'\ b$
8: $r'\ {}_3\# \rightarrow p\ \#\#$
9: $p\ {}_1\#\# \rightarrow f\ b$
10: $f\ {}_1\#\# \rightarrow f\ b$

For instance, H computes $baabaabaab = b(a^2b)^3$ as

$s\# \Rightarrow s'\#\#a\#\#$ [1] $\Rightarrow s'\#\#aa\#\#$ [2] $\Rightarrow p\#\#aa\#\#$ [3]
$\Rightarrow q\#a\#a\#\#$ [4] $\Rightarrow p\#a\#a\#a\#$[5]
$\Rightarrow q\#aa\#\#a\#$ [4] $\Rightarrow p\#aa\#\#aa\#$ [5]
$\Rightarrow r\#aa\#\#aa\#$ [6] $\Rightarrow r'baa\#\#aa\#$ [7] $\Rightarrow pbaa\#\#aa\#\#$ [8]
$\Rightarrow qbaa\#a\#a\#\#$ [4] $\Rightarrow pbaa\#a\#a\#a\#$ [5]
$\Rightarrow qbaa\#aa\#\#a\#$ [4] $\Rightarrow pbaa\#aa\#\#aa\#$ [5]
$\Rightarrow rbaa\#aa\#\#aa\#$ [6] $\Rightarrow r'baabaa\#\#aa\#$ [7] $\Rightarrow pbaabaa\#\#aa\#\#$ [8]
$\Rightarrow fbaabaabaa\#\#$ [9] $\Rightarrow fbaabaabaab$ [10].

It is obvious that H is of index 4 and $L(H) = \{b(a^ib)^j \mid i, j \geq 1\} \in \mathcal{L}_4(G\#RS$ with unrestricted rules). Thus, this example shows that for every $k \geq 1$, there is a language L such that $L \notin \mathcal{L}_k(CF\#RS)$ and L is generated by a generalized #-rewriting system with unrestricted rules.

We believe that there is no infinite hierarchy based on finite index in case of generalized #-rewriting systems with unrestricted rules. More specifically, every generalized #-rewriting system with unrestricted rules can be transformed into equivalent generalized #-rewriting system with unrestricted rules of index 1. However, we have not proved this conjecture so far.

Unlimited Index. Consider #-rewriting systems that are of unlimited index. What is the language family defined by them. What is the relation between $\mathcal{L}(CF\#RS)$ and $\mathcal{L}(G\#RS)$?

Determinism. This paper has discussed a general version of #-rewriting systems, which work non-deterministically. Of course, these systems can be restricted so they work deterministically. What is the language family defined by these deterministic systems?

Formally, a generalized #-rewriting system M is *deterministic* if for every left-hand side of M's rule r, $lhs(r)$, holds that $card(\{lhs(r) \mid r \in R\}) \leq 1$.

Acknowledgement. This work was supported by the Czech Science Foundation 201/07/0005 grant.

References

1. H. Borodihn, H. Fernau, Accepting grammars and systems: an overview. In: *Proc. of Development in Language Theory Conf.*, Magdeburg, 1995, pp. 199-208.
2. J. Dassow, G. Păun, *Regulated Rewriting in Formal Language Theory.* Springer, New York, 1989, 308 p., ISBN 0-38751-414-7.
3. T. Kasai, A Hierarchy Between Context-Free and Context-Sensitive Languages. In: *Journal of Computer and System Sciences* **4**, 1970, pp. 492-508.
4. 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.
5. A. Meduna, *Automata and Languages: Theory and Applications.* Springer, London, 2000, 916 p., ISBN 1-85233-074-0.
6. E. Moriya, D. Hofbauer, M. Huber, F. Otto, On state-alternating context-free grammars. In: *Theoretical Computer Science* **337**, 2005, p. 183-216.
7. G. Rozendberg, A. Salomaa (eds.), *Handbook of Formal Languages: Word, Language, Grammar*, Volume 1. Springer, Berlin, 1997, 873 p., ISBN 3-540-60420-0.