ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume 21, Number 3, 2018, 278–287

On k#\$-Rewriting Systems

Jirí Kucera[1], Zbynek Krivka[1], and Alexander Meduna[1]

[1]Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Božetěchova 2, 612 66 Brno, Czech Republic
Email: {ikucera, krivka, meduna}@fit.vutbr.cz

Abstract. This paper introduces k#\$-rewriting systems based on earlier defined #-rewriting systems but with additional pushdown memory. It demonstrates that these systems characterize an infinite hierarchy of language families resulting from the limited number of rewriting positions in every configuration during the generation of a sentence.

Key-words: k#\$-rewriting systems, pushdown, #-rewriting systems, infinite hierarchy, finite index, n-limited state grammars

1. Introduction

The most used classes of formal models in the formal language theory are grammars and automata. Grammars work as generative devices, while automata work as accepting devices. Given a grammar, it uses its rules to derive the string belonging to the language it describes from some initial string. Given an automaton, it uses its rules to decide which actions should be performed, based on its state, first symbol of its input string, and possibly on other additional information. Every string that drives the given automaton to its accepting configuration belongs to the language characterized by that automaton.

Formal language theory has introduced several language-defining models, *state grammars* and *deep pushdown automata* (see [1], [2]), based upon a combination of grammars and automata. To give a more general example, *rewriting systems* underlie both grammars and automata, so they generalize both. Furthermore, a unified approach to the way both grammars and automata define languages were studied in [3] and [4].

In 2006, Meduna, Křivka, and Schönecker introduced a new modification of rewriting systems, called *#-rewriting systems* (see [5]). While ordinary rewriting systems rewrite just one substring to another during one computation step, #-rewriting systems rewrite in fact two substrings, where the first substring is always one symbol long and acts like *state*. Moreover, the success of one computation step in #-rewriting systems depends also on the number of occurrences of # in their sentential forms. If k is an upper bound limit of the number of occurrences of #, #-rewriting systems are said to be of *index k*. Such a restriction has an important influence to their descriptive power. While ordinary rewriting systems characterize the Chomsky hierarchy of languages, the power of #-rewriting systems of index k coincide with the power of programmed grammars of the same index (see [5]).

In this paper, we extend #-rewriting systems with additional storage that can contain both terminals and nonterminals and we call them *k#\$-rewriting systems.* More precisely, every configuration consists of three parts: (1) the current state, (2) a string of terminals (including #), and, newly, (3) a pushdown string of terminals and nonterminals (excluding #). Below, in the string representation of such configuration, part (2) and (3) are separated by the symbol \$.

From a more practical viewpoint, it is noteworthy that this string representation is applicable in theoretically oriented computational linguistics related to the fundamental study given in [6], discussing pumping lemmas for the control language hierarchy. As a matter of fact, it can be used in any practically limited scattered information processing, such as bioinformatics, including state grammar and deep pushdown automata applied to biological sequences of nucleic acids (see [7]).

After giving some preliminaries in Section 2. and introducing the formal definition with an example in Section 3., in Section 4., we show that for some positive integer, k, k#\$-rewriting systems and k-limited state grammars have the same expressive power. Finally, the concluding section outlines an open problem for further investigation.

2. Preliminaries

This paper assumes that the reader is familiar with the fundamental notions of formal language theory (see [8, 9]). For a set X, $\mathrm{card}(X)$ denotes its cardinality and 2^X denotes its power set. By $\mathbb{N}$, we denote a set of all positive integers. Let Σ be an alphabet. Then, Σ^* represents the free monoid generated by Σ under the operation of concatenation with ε as its identity element. Set $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. For $w \in \Sigma^*$, $|w|$ denotes the length of w, $\mathrm{alph}(w) = \{x \mid w = uxv, x \in \Sigma, u, v \in \Sigma^*\}$ denotes the minimal subset of Σ such that $w \in \mathrm{alph}(w)^*$. For $a \in \Sigma$, $\mathrm{occur}(w, a)$ denotes the number of occurrences of a in w; mathematically, $\mathrm{occur}(w, a) = \mathrm{card}(\{u \mid w = uav, u, v \in \Sigma^*\})$. For $W \subseteq \Sigma$, $\mathrm{occur}(w, W) = \sum_{a \in W} \mathrm{occur}(w, a)$. For $k \geq 0$, if w can be expressed as $w = xy$ such that $k = |x|$ and $x, y \in \Sigma^*$, then $\mathrm{prefix}(w, k) = x$; otherwise, $\mathrm{prefix}(w, k) = w$.

Let A be a set and let σ be a (binary) relation over A. The k-fold product of σ, where $k \geq 0$, the transitive closure of σ, and the reflexive and transitive closure of σ are denoted as σ^k, σ^+, and σ^*, respectively. Instead of $(x, y) \in \sigma$, we write $x\ \sigma\ y$.

By $p\colon e$, we express that e has p as its *label*, i.e. p is a unique symbol that is associated with e and that can be used as an alternative name of e. By $p\colon e \in D$, we express that $p\colon e$ and $e \in D$.

A *context-free grammar* is a quadruple, $G = (V, T, P, S)$, where V is a total alphabet, $T \subset V$ is an alphabet of terminals, $P \subseteq (V - T) \times V^*$ is a finite set of rules, and $S \in (V - T)$ is the start symbol. Instead of $(A, x) \in P$, we write $A \to x \in P$. Let $\Rightarrow$ be a relation of direct derivation on V^* defined as follows: $uAv \Rightarrow uxv$ iff $A \to x \in P$, where $A \in (V - T)$ and $u, x, v \in V^*$. By $uAv \Rightarrow uxv\,[A \to x]$, we express that uAv

directly derives uxv according to $A \to x$. By $\Rightarrow_G$, we express that a relation of direct derivation, $\Rightarrow$, is associated with a grammar G. The language generated by G, $L(G)$, is defined as $L(G) = \{w \mid S \Rightarrow^* w, w \in T^*\}$. The family of context-free languages is denoted as $\mathscr{L}(\mathrm{CF})$.

Let $n \geq 1$ and $\Sigma_n = \{a_1, b_1, a_2, b_2, \ldots, a_n, b_n\}$. The *Dyck language* $\mathscr{D}_n$ over Σ_n is generated by the grammar

$$(\{S\} \cup \Sigma_n, \Sigma_n, \{S \to SS, S \to \varepsilon, S \to a_1 S b_1, \ldots, S \to a_n S b_n\}, S).$$

Let G be a grammar of arbitrary type, and let V, T, and S be its total alphabet, terminal alphabet, and start symbol, respectively. For a derivation $D\colon w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_r$, $S = w_1$, $w_r \in T^*$, according to G, we set $\mathrm{Ind}(D, G) = \max\{\mathrm{occur}(w_i, V - T) \mid 1 \leq i \leq r\}$, and for $w \in T^*$, we define $\mathrm{Ind}(w, G) = \min\{\mathrm{Ind}(D, G) \mid D \text{ is a derivation for } w \text{ in } G\}$. The *index of grammar* G (see page 151 in [10]) is defined as $\mathrm{Ind}(G) = \sup\{\mathrm{Ind}(w, G) \mid w \in L(G)\}$. For a language L in the family $\mathscr{L}(X)$ of languages generated by grammars of some type X, we define $\mathrm{Ind}_X(L) = \inf\{\mathrm{Ind}(G) \mid L(G) = L, G \text{ is of type } X\}$. For a family $\mathscr{L}(X)$, we set $\mathscr{L}_n(X) = \{L \mid L \in \mathscr{L}(X) \text{ and } \mathrm{Ind}_X(L) \leq n\}$, $n \geq 1$.

A *state grammar* (see [1]) is a sixtuple $G = (V, T, K, P, S, s)$, where V is a total alphabet, $T \subset V$ is an alphabet of terminals, K is a finite set of states, $V \cap K = \emptyset$, $P \subseteq (V - T) \times K \times V^* \times K$ is a finite set of rules, $S \in (V - T)$ is the start symbol, and $s \in K$ is the start state. Instead of $(A, p, x, q) \in P$, we write $(A, p) \to (x, q) \in P$. Let $\Rightarrow$ be a relation of direct derivation on $V^* \times K$ defined as follows: $(uAv, p) \Rightarrow (uxv, q)$ iff $(A, p) \to (x, q) \in P$ and for every $(B, p) \to (y, t) \in P$, $B \notin \mathrm{alph}(u)$, where $p, q, t \in K$, $A, B \in (V - T)$, and $u, v, x, y \in V^*$. For some $k \geq 1$ satisfying $\mathrm{occur}(uA, V - T) \leq k$, $\Rightarrow$ is said to be k-limited, denoted as ${}_k\!\Rightarrow$. By $(uAv, p) \Rightarrow (uxv, q)\,[(A, p) \to (x, q)]$, we express that (uAv, p) directly derives (uxv, q) according to $(A, p) \to (x, q)$. Similarly for ${}_k\!\Rightarrow$. The language generated by G, $L(G)$, is defined as $L(G) = \{w \mid (S, s) \Rightarrow^* (w, q), q \in K, w \in T^*\}$. Let $k \geq 1$. The language generated by G in k-limited way, $L(G, k)$, is defined as $L(G, k) = \{w \mid (S, s)\ {}_k\!\Rightarrow^* (w, q), q \in K, w \in T^*\}$. The families of languages generated by state grammars and by state grammars in k-limited way are denoted as $\mathscr{L}(\mathrm{ST})$ and $\mathscr{L}(\mathrm{ST}, k)$, respectively.

A *#-rewriting system* (see [5]) is a quadruple $M = (Q, \Sigma, s, R)$, where Q is a finite set of states, Σ is an alphabet containing special symbol # called *bounder*, $Q \cap \Sigma = \emptyset$, $s \in Q$ is the start state, and $R \subseteq Q \times \mathbb{N} \times \{\#\} \times Q \times \Sigma^*$ is a finite set of rules. Instead of $(p, n, \#, q, x) \in R$, we write $p\,{}_n\# \to qx$. Let $\Rightarrow$ be a relation of direct rewriting step on $Q\Sigma^*$ defined as follows: $pu\#v \Rightarrow quxv$ iff $p\,{}_n\# \to qx \in R$ and $\mathrm{occur}(u, \#) = n - 1$, where $p, q \in Q$, $u, v, x \in \Sigma^*$, and $n \in \mathbb{N}$. By $pu\#v \Rightarrow quxv\,[p\,{}_n\# \to qx]$, we express that $pu\#v$ directly rewrites $quxv$ according to $p\,{}_n\# \to qx$. The language generated by M, $L(M)$, is defined as $L(M) = \{w \mid s\# \Rightarrow^* qw, q \in Q, w \in (\Sigma - \{\#\})^*\}$. Let $k \in \mathbb{N}$. A #-rewriting system M is said to be of index k iff $s\# \Rightarrow^* qy$ implies $\mathrm{occur}(y, \#) \leq k$, where $q \in Q$ and $y \in \Sigma^*$. Let $k \in \mathbb{N}$. The family of languages generated by #-rewriting systems and by #-rewriting systems of index k are denoted as $\mathscr{L}(\#\mathrm{RS})$ and $\mathscr{L}_k(\#\mathrm{RS})$, respectively.

3. Definitions

We are now ready to define $k\#\$$-rewriting systems.

Definition 3..1. *Let $k \in \mathbb{N}$. A $k\#\$$-*rewriting system *is a quintuple*

$$M = (Q, V, \Sigma, s, R),$$

where Q is a finite set of states, V is a total alphabet, $V \cap Q = \emptyset$, Σ is an alphabet containing $\#$ and $\$$ called bounders, *$\Sigma \subseteq V$, $s \in Q$ is a start state, and*

$$\begin{array}{rcl} R & \subseteq & (Q \times \mathbb{N} \times \{\#\} \times Q \times (\Sigma - \{\$\})^*) \\ & \cup & (Q \times \{\#\} \times \{\$\} \times Q \times \{\$\} \times (V - \{\#, \$\})^*) \\ & \cup & (Q \times \{\$\} \times (V - \Sigma) \times Q \times \{\#\} \times \{\$\}) \end{array}$$

is a finite set of rules.

Instead of $(p, n, \#, q, x) \in R$, $(p, \#, \$, q, \$, y) \in R$, and $(p, \$, A, q, \#, \$) \in R$, we write $p\,{}_n\# \to qx \in R$, $p\#\$ \to q\$y \in R$, and $p\$A \to q\#\$ \in R$, respectively.

Let $\Xi \subseteq Q(\Sigma - \{\$\})^*\{\$\}(V - \{\#, \$\})^*$ be a set of all configurations *of M such that $\chi \in \Xi$ iff* $\mathrm{occur}(\chi, \#) \le k$.

Let $\Rightarrow$ be a relation of direct rewriting step on Ξ defined as follows:

- *$pu\#v\$\alpha \Rightarrow quxv\$\alpha$ iff $p\,{}_n\# \to qx \in R$, $\mathrm{occur}(u, \#) = n - 1$, $p, q \in Q$, $u, v, x \in (\Sigma - \{\$\})^*$, $\alpha \in (V - \{\#, \$\})^*$, and $n \in \mathbb{N}$;*
- *$pu\#\$\alpha \Rightarrow qu\$x\alpha$ iff $p\#\$ \to q\$x \in R$, $p, q \in Q$, $u \in (\Sigma - \{\$\})^*$, and $x, \alpha \in (V - \{\#, \$\})^*$;*
- *$pu\$A\alpha \Rightarrow qu\#\$\alpha$ iff $p\$A \to q\#\$ \in R$, $p, q \in Q$, $u \in (\Sigma - \{\$\})^*$, $A \in V - \Sigma$, and $\alpha \in (V - \{\#, \$\})^*$;*
- *$pux\$\alpha \Rightarrow pu\$x\alpha$ iff $p \in Q$, $u \in (\Sigma - \{\$\})^*$, $x \in (\Sigma - \{\#, \$\})^*$, and $\alpha \in (V - \{\#, \$\})^*$;*
- *$pu\$x\alpha \Rightarrow pux\$\alpha$ iff $p \in Q$, $u \in (\Sigma - \{\$\})^*$, $x \in (\Sigma - \{\#, \$\})^*$, and $\alpha \in (V - \{\#, \$\})^*$.*

By $x \Rightarrow y\,[r]$, we express that x directly rewrites y according to r.

The language generated by M, $L(M)$, is defined as

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

The family of languages generated by $k\#\$$-rewriting systems is denoted as $\mathscr{L}_k(\#\$\mathrm{RS})$.

The following example demonstrates a generative capacity of $k\#\$$-rewriting systems.

Example 3..2. *Let $M = (Q, V, \Sigma, s, R)$ be a 2#$-rewriting system, where*

$$\begin{array}{rcl} Q & = & \{s, p, p', p^{(1)}, p^{(2)}, p^{(X)}, p^{(Y)}, q, f, f^{(A)}, f^{(B)}\} \\ V & = & \{A, B, X, a, b, c, d, 0, 1, \bar{0}, \bar{1}, [_1, [_2,]_1,]_2, \#, \$\} \\ \Sigma & = & \{a, b, c, d, 0, 1, \bar{0}, \bar{1}, [_1, [_2,]_1,]_2, \#, \$\} \end{array}$$

and R contains rules

$$\begin{array}{rlrl} 1: & s\,{}_1\# \to p\#\# & 9: & p^{(Y)}\,{}_1\# \to q \\ 2: & p\,{}_1\# \to p'a\#b & 10: & q\,{}_1\# \to f \\ 3: & p'\,{}_2\# \to p^{(1)}c\# & 11: & f\$A \to f^{(A)}\#\$ \\ 4: & p'\,{}_2\# \to p^{(2)}d\# & 12: & f\$B \to f^{(B)}\#\$ \\ 5: & p^{(1)}\#\$ \to p^{(X)}\$X[_1A]_1 & 13: & f^{(A)}\,{}_1\# \to f^{(A)}0\#1 \\ 6: & p^{(2)}\#\$ \to p^{(X)}\$X[_2B]_2 & 14: & f^{(B)}\,{}_1\# \to f^{(B)}\bar{0}\#\bar{1} \\ 7: & p^{(X)}\$X \to p\#\$ & 15: & f^{(A)}\,{}_1\# \to f01 \\ 8: & p^{(X)}\$X \to p^{(Y)}\#\$ & 16: & f^{(B)}\,{}_1\# \to f\bar{0}\bar{1} \end{array}$$

First, M generates two $\#$ bounders. Second, M uses rules 2 through 7 to generate the following structure

$$a^m\#b^m z_1z_2\cdots z_m\#\$\phi(z_mz_{m-1}\cdots z_1)$$

where $z_i\in\{c,d\}$, $1\le i\le m$, $m\ge 1$ and ϕ is a homomorphism from $\{c,d\}^*$ to $\{A,B,[_1,[_2,]_1,]_2\}^*$ such that $\phi(c)=[_1A]_1$ and $\phi(d)=[_2B]_2$. Finally, M uses rules 8 through 16 to finish the rewriting. Thus, the language generated by M is

$$L(M)=\left\{w \;\middle|\; \begin{array}{l} w=a^nb^nz_1z_2\ldots z_nh(z_n,i_1)h(z_{n-1},i_2)\ldots h(z_1,i_n), \\ z_i\in\{c,d\}, 1\le i\le n, i_j\ge 1, 1\le j\le n, n\ge 1 \end{array}\right\}$$

where h is a mapping from $\{c,d\}\times\mathbb{N}$ to $\{0,1,\bar{0},\bar{1},[_1,[_2,]_1,]_2\}^*$ such that $h(c,i)=[_10^i1^i]_1$ and $h(d,i)=[_2\bar{0}^i\bar{1}^i]_2$.

For instance, M generates $aabbdc[_10011]_1[_2\bar{0}\bar{1}]_2$ in the following way

$$\begin{array}{rlll}
s\#\$ & \Rightarrow & p\#\#\$ & [1] \\
 & \Rightarrow & p'a\#b\#\$ & [2] \\
 & \Rightarrow & p^{(2)}a\#bd\#\$ & [4] \\
 & \Rightarrow & p^{(X)}a\#bd\$X[_2B]_2 & [6] \\
 & \Rightarrow & pa\#bd\#\$[_2B]_2 & [7] \\
 & \Rightarrow & p'aa\#bbd\#\$[_2B]_2 & [2] \\
 & \Rightarrow & p^{(1)}aa\#bbdc\#\$[_2B]_2 & [3] \\
 & \Rightarrow & p^{(X)}aa\#bbdc\$X[_1A]_1[_2B]_2 & [5] \\
 & \Rightarrow & p^{(Y)}aa\#bbdc\#\$[_1A]_1[_2B]_2 & [8] \\
 & \Rightarrow & qaabbdc\#\$[_1A]_1[_2B]_2 & [9] \\
 & \Rightarrow & faabbdc\$[_1A]_1[_2B]_2 & [10] \\
 & \Rightarrow & faabbdc[_1\$A]_1[_2B]_2 & \\
 & \Rightarrow & f^{(A)}aabbdc[_1\#\$]_1[_2B]_2 & [11] \\
 & \Rightarrow & f^{(A)}aabbdc[_10\#1\$]_1[_2B]_2 & [13] \\
 & \Rightarrow & faabbdc[_10011\$]_1[_2B]_2 & [15] \\
 & \Rightarrow & faabbdc[_10011]_1[_2\$B]_2 & \\
 & \Rightarrow & f^{(B)}aabbdc[_10011]_1[_2\#\$]_2 & [12] \\
 & \Rightarrow & faabbdc[_10011]_1[_2\bar{0}\bar{1}\$]_2 & [16] \\
 & \Rightarrow & faabbdc[_10011]_1[_2\bar{0}\bar{1}]_2\$ &
\end{array}$$

4. Results

First, we prove the identity of $\mathscr{L}(\mathrm{ST},k)$ and $\mathscr{L}_k(\#\$\mathrm{RS})$ for every $k\ge 1$.

Lemma 4..1. *Let $k\ge 1$. Then, $\mathscr{L}(\mathrm{ST},k)\subseteq\mathscr{L}_k(\#\$\mathrm{RS})$.*

Proof. Let $G=(V,T,K,P,S,s)$ be a state grammar. Without any loss on generality, suppose that $V\cap\{\#,\$\}=\emptyset$. Now, we construct from G a $k\#\$$-rewriting system

$$M=(Q,V',\Sigma,s',R)$$

such that $L(G,k) = L(M)$. First, we set

$$\begin{array}{rcl} Q & = & \bigcup_{i=0}^{k}\{\langle q;l;u\rangle \mid q \in K, u \in (V-T)^i, 0 \le l \le k\} \\ V' & = & V \cup \{\#, \$\} \\ \Sigma & = & T \cup \{\#, \$\} \\ s' & = & \langle s;0;S\rangle \end{array}$$

In Q, each state records three pieces of information—(1) the current state of G, (2) auxiliary substate of the simulation (0 is for regular state, and 1 through k are for auxiliary substates), and (3) the first k nonterminal symbols from the current sentential form of G. The positions of these k symbols correspond to #s in the simulation of G by M.

Next, we construct R. Let

$$\text{rules}(p,u) = \left\{ r \,\middle|\, \begin{array}{l} r\colon (B,p) \to (x,q) \in P, B \in ((V-T) \cap \text{alph}(u)), \\ p,q \in K, x \in V^+, u \in V^* \end{array} \right\}$$

and let g and h be two homomorphisms from V^* to $(\Sigma - \{\$\})^*$ and from V^* to $(V' - \Sigma)^*$, respectively, defined as

$$\begin{array}{rcl} g(x) & = & \left\{ \begin{array}{ll} \# & \text{for every } x \in (V-T) \\ x & \text{for every } x \in T \end{array} \right. \\ h(x) & = & \left\{ \begin{array}{ll} x & \text{for every } x \in (V-T) \\ \varepsilon & \text{for every } x \in T \end{array} \right. \end{array}$$

Initially, set $R = \emptyset$. For every rule $(A,p) \to (x,q) \in P$ and for every state $\langle p;0;uAv\rangle \in Q$ such that $\text{rules}(p,u) = \emptyset$ perform the following steps:

(A) If $k - |uv| \ge |h(x)|$, then add $\langle p;0;uAv\rangle\,_{|uA|}\# \to \langle q;0;uh(x)v\rangle g(x)$ to R.

(B) If $k - |uv| < |h(x)|$, then express v as $v = X_{m-1}X_{m-2}\cdots X_0$, where $X_i \in (V' - \Sigma)$, $0 \le i \le m-1$, $m = |v|$, and

  (i) for every i such that $0 \le i \le m-1$, add $\langle p;i;uAv\rangle\#\$ \to \langle p;i+1;uAv\rangle\$X_i$ to R;

  (ii) add $\langle p;m;uAv\rangle\#\$ \to \langle q;0;u\rangle\$x$ to R.

Finally, for every state $\langle p;0;u\rangle \in Q$ such that $|u| \le k-1$ and for every $B \in (V' - \Sigma)$ add rule

$$\langle p;0;u\rangle\$B \to \langle p;0;uB\rangle\#\$$$

to R. The construction of M is completed.

Due to the lack of space, we leave the formal proof that $L(G,k) = L(M)$, which is rather technical, to the kind reader. Both, $L(G,k) \subseteq L(M)$ and $L(M) \subseteq L(G,k)$ can be proved by induction on the number of derivation or rewriting steps, respectively.

□

Lemma 4..2. *Let $k \ge 1$. Then, $\mathscr{L}_k(\#\$\text{RS}) \subseteq \mathscr{L}(\text{ST}, k)$.*

Proof. Let $M = (Q, V, \Sigma, s, R)$ be a $k\#\$$-rewriting system. Without any loss on generality, suppose that $¿ \notin V$ and $\#_i \notin V$, for all $1 \leq i \leq k$. From M, we construct a state grammar

$$G = (V', T, K, P, \#_1, s')$$

such that $L(M) = L(G, k)$. First, we set

$$\begin{array}{rcl} V' & = & (V - \{\#, \$\}) \cup \{\#_i \mid 1 \leq i \leq k\} \\ T & = & \Sigma - \{\#, \$\} \\ K & = & \{\langle p; i\rangle \mid p \in Q, 0 \leq i \leq k\} \cup \{\langle p; i; r\rangle \mid p \in Q, 0 \leq i \leq k, r \in R\} \\ & \cup & \{\langle p; i; [\![r, j]\!]\rangle \mid p \in Q, r \in R, 0 \leq i \leq k, 1 \leq j \leq k\} \\ & \cup & \{\langle p; i; [\![r, X]\!]\rangle \mid p \in Q, r \in R, X \in (V - \Sigma) \cup \{¿\}, 0 \leq i \leq k\} \\ & \cup & \{q_{\text{fail}}\} \\ s' & = & \langle s; 1\rangle \end{array}$$

In K, each state records the current state of M and the number of #s in the current configuration of M. Sometimes, it also contains the simulated rule, r, and additionally with information either about the leftmost non-# nonterminal symbol, X, or simulation progress, j. Note that $X = ¿$ if the simulation of a rule from R of the form $r\colon p\$A \to q\#\$$ has started. In addition, K contains a special state q_{fail} that brings G to a configuration that rules out the next derivation step in G, which unsuccessfully stops the simulation of M.

Let τ be a mapping from $(\Sigma - \{\$\})^* \times \{1, 2, \ldots, k\}$ to $(T \cup \{\#_i \mid 1 \leq i \leq k\})^*$ defined recursively as follows

- $\tau(\varepsilon, i) = \varepsilon$, for every $1 \leq i \leq k$
- $\tau(ax, i) = a\tau(x, i)$, for every $a \in (\Sigma - \{\#, \$\})$, $x \in (\Sigma - \{\$\})^*$, and $1 \leq i \leq k$
- $\tau(\#x, i) = \#_i\tau(x, i + 1)$, for every $x \in (\Sigma - \{\$\})^*$ and $1 \leq i \leq k - 1$

We are now ready to construct P. Initially, set $P = \emptyset$. For every state $\langle p; \kappa\rangle \in K$ and for every rule $r\colon p\,{}_n\# \to qx \in R$ such that $n \leq \kappa$ and $\kappa - 1 + \text{occur}(x, \#) \leq k$ perform the following steps:

(A) If $\text{occur}(x, \#) = 0$ and $\kappa - n = 0$, then add

$$\begin{array}{rcl} (\#_1, \langle p; \kappa\rangle) & \to & (\#_1, \langle p; \kappa; r\rangle) \\ (\#_\kappa, \langle p; \kappa; r\rangle) & \to & (x, \langle q; \kappa - 1\rangle) \end{array}$$

to P.

(B) If $\text{occur}(x, \#) = 0$ and $\kappa - n \geq 1$, then

- add $(\#_1, \langle p; \kappa\rangle) \to (\#_1, \langle p; \kappa; r\rangle)$ to P;
- add $(\#_n, \langle p; \kappa; r\rangle) \to (x, \langle q; \kappa - 1; [\![r, 1]\!]\rangle)$ to P;
- for every $1 \leq i \leq \kappa - n - 1$, add

$$(\#_{n+i}, \langle q; \kappa - 1; [\![r, i]\!]\rangle) \to (\#_{n+i-1}, \langle q; \kappa - 1; [\![r, i + 1]\!]\rangle)$$

  to P;

- add $(\#_\kappa, \langle q; \kappa-1; [\![r, \kappa-n]\!]\rangle) \to (\#_{\kappa-1}, \langle q; \kappa-1\rangle)$ to P.

(C) If $\mathrm{occur}(x, \#) = 1$, then add

$$\begin{array}{rcl} (\#_1, \langle p; \kappa\rangle) & \to & (\#_1, \langle p; \kappa; r\rangle) \\ (\#_n, \langle p; \kappa; r\rangle) & \to & (\tau(x, n), \langle q; \kappa\rangle) \end{array}$$

to P.

(D) If $\mathrm{occur}(x, \#) \geq 2$, then

- add $(\#_1, \langle p; \kappa\rangle) \to (\#_1, \langle p; \kappa; r\rangle)$ to P;
- add $(\#_n, \langle p; \kappa; r\rangle) \to (\#_n, \langle p; \kappa; [\![r, 1]\!]\rangle)$ to P;
- for every $0 \leq i \leq \kappa - n - 1$, add

$$(\#_{\kappa-i}, \langle p; \kappa; [\![r, i+1]\!]\rangle) \to (\#_{\kappa+\eta-i}, \langle p; \kappa; [\![r, i+2]\!]\rangle)$$

  to P, where $\eta = \mathrm{occur}(x, \#) - 1$;
- add $(\#_n, \langle p; \kappa; [\![r, \kappa-n+1]\!]\rangle) \to (\tau(x, n), \langle q; \kappa+\eta\rangle)$ to P, where $\eta = \mathrm{occur}(x, \#) - 1$.

Next, for every state $\langle p; \kappa\rangle \in K$ such that $\kappa \geq 1$ and for every rule $r\colon p\#\$ \to q\$x \in R$, add

$$\begin{array}{rcl} (\#_1, \langle p; \kappa\rangle) & \to & (\#_1, \langle p; \kappa; r\rangle) \\ (\#_\kappa, \langle p; \kappa; r\rangle) & \to & (x, \langle q; \kappa-1\rangle) \end{array}$$

to P.

Finally, for every state $\langle p; \kappa\rangle \in K$ such that $\kappa \leq k-1$ and for every rule $r\colon p\$A \to q\#\$ \in R$, add

- $(A, \langle p; \kappa\rangle) \to (A, \langle p; \kappa; r\rangle)$ if $\kappa = 0$
- $(\#_1, \langle p; \kappa\rangle) \to (\#_1, \langle p; \kappa; r\rangle)$ if $\kappa \geq 1$
- $(A, \langle p; \kappa; r\rangle) \to (A, \langle p; \kappa; [\![r, ¿]\!]\rangle)$
- $(X, \langle p; \kappa; [\![r, ¿]\!]\rangle) \to (X, \langle p; \kappa; [\![r, X]\!]\rangle)$, for all $X \in (V - \Sigma)$
- $(Y, \langle p; \kappa; [\![r, Y]\!]\rangle) \to (Y, q_{\mathrm{false}})$, for all $Y \in (V - \Sigma)$, where $Y \neq A$
- $(A, \langle p; \kappa; [\![r, A]\!]\rangle) \to (\#_{\kappa+1}, \langle q; \kappa+1\rangle)$

to P. Now, the construction of G is completed.

As in the proof of Lemma 4..1, the formal proof of Lemma 4..2 has a similar structure and its technical details are left out.

□

Corollary 4..3. *Let $k \geq 1$. Then, $\mathscr{L}_k(\#\$\mathrm{RS}) = \mathscr{L}(\mathrm{ST}, k)$.*

Proof. It directly follows from Lemma 4..1 and Lemma 4..2. □

Next, we show that $\mathscr{L}_k(\#\mathrm{RS})$ is properly included in $\mathscr{L}_k(\#\$\mathrm{RS})$ for every $k \geq 1$.

Theorem 4..4. *For every $k \geq 1$. Then, $\mathscr{L}_k(\#\mathrm{RS}) \subset \mathscr{L}_k(\#\$\mathrm{RS})$.*

Proof. The inclusion $\mathscr{L}_k(\#\mathrm{RS}) \subseteq \mathscr{L}_k(\#\$\mathrm{RS})$ follows directly from the definitions of #-rewriting system of index k and k#$-rewriting system. It remains to find a language from $\mathscr{L}_k(\#\$\mathrm{RS})$ that is not contained in $\mathscr{L}_k(\#\mathrm{RS})$.

For $k = 1$, such a language is $\mathscr{D}_2$. As $\mathscr{L}_1(\#\$\mathrm{RS}) = \mathscr{L}(\mathrm{CF})$ (by [1] and Corollary 4..3), $\mathscr{D}_2 \in \mathscr{L}_1(\#\$\mathrm{RS})$, but $\mathscr{D}_2 \notin \mathscr{L}_1(\#\mathrm{RS})$ (see page 169 in [10]).

For $k \geq 2$, let $\Sigma_k = \{a_1, a_2, \dots, a_{4k-2}\}$ be an alphabet. Define a language L_k over Σ_k as

$$L_k = \{a_1^i a_2^i \dots a_{4k-2}^i \mid i \geq 1\}.$$

By Theorem 4 in [1], $L_k \in \mathscr{L}(\mathrm{ST}, k)$ and since $\mathscr{L}_k(\#\$\mathrm{RS}) = \mathscr{L}(\mathrm{ST}, k)$, $L_k \in \mathscr{L}_k(\#\$\mathrm{RS})$ as well.

It is easy to see that matrix grammars of finite index k generates the same language family as $\mathscr{L}_k(\#\mathrm{RS})$ (see [5] and Theorem 3.1.2 on page 155 in [10]). By an application of pumping lemma for matrix grammars of finite index (see Lemma 3.1.6 on page 159 in [10]), we can prove that $L_k \notin \mathscr{L}_k(\#\mathrm{RS})$. Assume that $L_k \in \mathscr{L}_k(\#\mathrm{RS})$. Therefore, there exists $z \in L_k$ such that

$$z = u_1 v_1 w_1 x_1 u_2 v_2 w_2 x_2 \dots u_l v_l w_l x_l u_{l+1}$$

with $l \leq k$, $|v_1 x_1 v_2 x_2 \dots v_l x_l| > 0$, and

$$u_1 v_1^i w_1 x_1^i u_2 v_2^i w_2 x_2^i \dots u_l v_l^i w_l x_l^i u_{l+1} \in L_k$$

for every $i \geq 1$. Now, consider the following cases:

- There exists $y \in \{v_1, x_1, v_2, x_2, \dots, v_l, x_l\}$ such that $\mathrm{card}(\mathrm{alph}(y)) \geq 2$. In this case, there exists $i \geq 1$ such that
$$u_1 v_1^i w_1 x_1^i u_2 v_2^i w_2 x_2^i \dots u_l v_l^i w_l x_l^i u_{l+1} \notin L_k.$$
- All $v_1, x_1, v_2, x_2, \dots, v_l, x_l$ are strings over one-letter alphabet. As for $k \geq 2$ it is always true that $4k - 2 > 2k$, there will be always symbols from $\mathrm{alph}(z)$ that are not contained in $\mathrm{alph}(v_1 x_1 v_2 x_2 \dots v_l x_l)$. Hence there must exist $i \geq 1$ such that
$$u_1 v_1^i w_1 x_1^i u_2 v_2^i w_2 x_2^i \dots u_l v_l^i w_l x_l^i u_{l+1} \notin L_k.$$

Such $z \in L_k$ does not exist and therefore $L_k \notin \mathscr{L}_k(\#\mathrm{RS})$ for every $k \geq 2$. □

The relationship between infinite hierarchies of #-rewriting systems of finite index and k#$-rewriting systems is summed in Figure 1.

5. Conclusion

In the future, we plan to investigate these three open problem areas.

(1) Since we have a new characterization of $\mathscr{L}(\mathrm{ST}, k)$, for some $k \geq 1$, we will study the relationship between k#$-rewriting systems and generalized #-rewriting systems (studied in Sections 4.1.4 and 5.1.3 of [11]).

(2) Considering [12], we intend to discuss the subject of this paper in terms of picture languages or 2D languages.

(3) Finally, we want to relate this paper to the study given in [13], in which multi-head finite automata are capable of simulating the tape portion behind \$ in k#\$-rewriting systems.

$$\begin{array}{ccccccc}
\mathscr{L}_1(\#\$\mathrm{RS}) & \subset & \mathscr{L}_2(\#\$\mathrm{RS}) & \subset & \cdots & \subset & \mathscr{L}_k(\#\$\mathrm{RS}) \\
\cup & & \cup & & & & \cup \\
\mathscr{L}_1(\#\mathrm{RS}) & \subset & \mathscr{L}_2(\#\mathrm{RS}) & \subset & \cdots & \subset & \mathscr{L}_k(\#\mathrm{RS})
\end{array}$$

Fig. 1. The relations between #-rewriting systems with finite index and #$-rewriting systems.

Acknowledgment. This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II), project IT4Innovations excellence in science – LQ1602; the TAČR grant TE01020415; and the BUT grant FIT-S-17-3964.

References

[1] T. Kasai, "An hierarchy between context-free and context-sensitive languages," *Journal of Computer and System Sciences*, vol. 4, no. 5, pp. 492–508, 1970.

[2] A. Meduna, "Deep pushdown automata," *Acta Informatica*, vol. 42, no. 8, pp. 541–552, 2006.

[3] P. Deussen, "A unified approach to the generation and the acception of formal languages," *Acta Informatica*, vol. 9, no. 4, pp. 377–390, 1978.

[4] A. Salomaa, *Computation and Automata.* Cambridge University Press, New York, 1985.

[5] Z. Křivka, A. Meduna, and R. Schönecker, "Generation of languages by rewriting systems that resemble automata," *International Journal of Foundations of Computer Science*, vol. 17, no. 5, pp. 1223–1229, 2006.

[6] M. A. Palis and S. M. Shende, "Pumping lemmas for the control language hierarchy," *Mathematical Systems Theory*, vol. 28, no. 3, pp. 199–213, 1995.

[7] N. Kalra and A. Kumar, "State grammar and deep pushdown automata for biological sequences of nucleic acids," *Current Bioinformatics*, vol. 11, no. 4, pp. 470–479, 2016.

[8] G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages: Word, Language, Grammar*, vol. 1. Berlin: Springer-Verlag, 1997.

[9] A. Salomaa, *Formal Languages.* Academic Press, London, 1973.

[10] J. Dassow and G. Păun, *Regulated Rewriting in Formal Language Theory.* Springer-Verlag, Berlin, 1989.

[11] Z. Křivka, *Rewriting Systems with Restricted Configurations.* Brno University of Technology, Faculty of Information Technology, 2008.

[12] R. Siromoney and K. G. Subramanian, "Selective substitution array grammars," *Information Sciences*, vol. 25, pp. 73–83, 1981.

[13] I. H. Sudborough, "On tape-bounded complexity classes and multi-head finite automata," *Journal of Computer and System Sciences*, vol. 10, no. 1, pp. 62–76, 1975.