DETERMINISTIC #-REWRITING SYSTEMS

Zbyněk Křivka
Doctoral Degree Programme (3), FIT BUT
E-mail: krivka@fit.vutbr.cz

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

ABSTRACT

This paper discusses a deterministic version of #-rewriting systems with context-free rules. It demonstrates that classical form of determinism does not affect the generative power of #-rewriting systems. The result concerning deterministic #-rewriting systems of index k is given too. The conclusion discusses even the stronger type of determinism.

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 $p\,{}_i\# \to q\,\gamma$, 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 deterministic version of #-rewriting systems that satisfy that system is deterministic if and only if there is no more than one rule in the set of rules with the same left-hand side (see Conclusion of [4]).

As its main result, this paper demonstrates that the determinism under discussion does not affect the generative power of #-rewriting systems. In terms of finite index, we are able to simulate a non-deterministic #-rewriting systems of index k by a deterministic #-rewriting systems of index $k+1$. Therefore, one additional bounder is needed in case of limited finite index.

This result is of some interest when compared, for instance, to a classical context-free models are much stronger than deterministic context-free grammars.

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 concatena-

tion. 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 $i \geq |w|+1$. $suffixes(w) = \{suffix(w,j) \mid 0 \leq j \leq |w|\}$. 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 $i \geq |w|+1$. $prefixes(w) = \{prefix(w,j) \mid 0 \leq j \leq |w|\}$.

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. For a set, Q, $card(Q)$ denotes the *cardinality* of Q. 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 $h \in K$, $k \leq h$.

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 $q{:}A \to v$, where $A \in (V-T)$, $v \in V^*$ and q is a label of this rule. If $q{:}A \to v \in P$, $x,y \in V^*$, G makes a derivation step from xAy to xvy according to $q{:}A \to v$, 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 $x \Rightarrow^m y\ [q_1q_2\ldots q_m]$. 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.

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.

3 DEFINITIONS

A *#-rewriting system* (abbr. CF#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 $R \subseteq Q \times I \times \{\#\} \times Q \times \Sigma^*$ is a finite relation whose members are called *rules*. A rule $(p,n,\#,q,x) \in R$, where $n \in I$, $q,p \in Q$ and $x \in \Sigma^*$, is usually written as $r{:}\, p_n\# \to q\ x$ 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, $p,q \in Q$, $u,v \in \Sigma^*$, $n \in I$ and $occur(u,\#) = n-1$. Then, H makes a *computational step* from $pu\#v$ to $quxv$ by using $r{:}\, p_n\# \to q\ x$, symbolically written $pu\#v \Rightarrow quxv\ [r]$ in M or simply $pu\#v \Rightarrow quxv$.

In the standard manner, we extend $\Rightarrow$ to $\Rightarrow^m$, for $m \geq 0$; then, based on $\Rightarrow^m$, we define $\Rightarrow^+$ and $\Rightarrow^*$ in the standard way. The *language generated* by M, $L(H)$, is defined as

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

Let k be a positive integer. A #-rewriting system 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.

As special case of CF#RS H, if for every $p \in Q$ and every positive integer i, $p_i\#$ is the left-hand side of no more than one rule in H, then H is called *deterministic #-rewriting system* (abbr. detCF#RS).

Let k be a positive integer. $\mathcal{L}(CF\#RS)$, $\mathcal{L}(detCF\#RS)$, $\mathcal{L}_k(CF\#RS)$, and $\mathcal{L}_k(detCF\#RS)$ denote the families of languages derived by #-rewriting systems, deterministic #-rewriting systems, #-rewriting systems of index k, and deterministic #-rewriting systems of index k, respectively.

Example 1. $H = (\{s, p, q, f\}, \{a, b, c, \#\}, s, R)$, where R 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: $f\ {}_1\# \rightarrow f\ c$

Obviously, H is of index 2 and non-deterministic (see rules 2 and 4), and $L(H) = \{a^n b^n c^n \mid n \geq 1\}$. For instance, H generates $aaabbbccc$ as $s\# \Rightarrow p\#\#$ [1] $\Rightarrow qa\#b\#$ [2] $\Rightarrow pa\#b\#c$ [3] $\Rightarrow qaa\#bb\#c$ [2] $\Rightarrow paa\#bb\#cc$ [3] $\Rightarrow faaabbb\#cc$ [4] $\Rightarrow faaabbbccc$ [5].

4 RESULTS

Theorem 1. $\mathcal{L}(CF\#RS) = \mathcal{L}(detCF\#RS)$.
Proof.
Since detCF#RS is a special case of CF#RS, we only need to prove that $\mathcal{L}(CF\#RS) \subseteq \mathcal{L}(detCF\#RS)$.
Construction.
Let $H = (Q_H, \Sigma, s_H, R_H)$ be a CF#RS, where $\Sigma = T \cup \{\#\}$. We construct the detCF#RS, $D = (Q_D, \Sigma, s_H, R_D)$, where R_D and Q_D are constructed by performing the following steps:

1. Let for every $p \in Q_H$ and $i \in I$, ${}^i_pR_H = \{r\colon p\ {}_i\# \rightarrow q\ x \mid r \in R_H, q \in Q_H, x \in \Sigma\}$;
2. Set $R' = \bigcup\{{}^i_pR_H \mid card({}^i_pR_H) \leq 2\}$;
3. For every ${}^i_pR_H$ with $card({}^i_pR_H) \geq 3$ do:
   $o := p$;
   `while` $(card({}^i_pR_H) \geq 3)$ `do:`
   - exclude r from ${}^i_pR_H$
   - add $\langle {}^i_pR_H \rangle$ into Q_D
   - add $o\ {}_i\# \rightarrow rhs(r)$ and $o\ {}_i\# \rightarrow \langle {}^i_pR_H \rangle\ \#$ into R';
   - $o := \langle {}^i_pR_H \rangle$;
4. Set $R_D = \bigcup\{{}^i_pR' \mid card({}^i_pR') = 1\}$.
5. Let h be a bijection from $lab(R')$ to I. For every pair of rules $r_i\colon p\ {}_i\# \rightarrow q_1\ x_1$ and $r_j\colon p\ {}_i\# \rightarrow q_2\ x_2$ from R', such that $h(i) < h(j)$, add into R_D the following rules:
   - $p\ {}_i\# \rightarrow \langle r_i, r_j \rangle\ \#\#$
   - $\langle r_i, r_j \rangle\ {}_i\# \rightarrow \langle r_i \rangle\ \#$
   - $\langle r_i, r_j \rangle\ {}_{i+1}\# \rightarrow \langle r_j \rangle\ \#$

- $\langle r_i \rangle \, {}_i\# \rightarrow \langle r'_i \rangle \, x_1$
- $\langle r_j \rangle \, {}_i\# \rightarrow \langle r'_j \rangle \, x_2$
- $\langle r'_i \rangle \, {}_{i+1}\# \rightarrow q_1 \, \varepsilon$
- $\langle r'_j \rangle \, {}_{i+1}\# \rightarrow q_2 \, \varepsilon$

and add new states $\langle r_i, r_j \rangle, \langle r_i \rangle, \langle r_i \rangle, \langle r'_i \rangle, \langle r'_j \rangle$ into Q_D.

Basic Idea.

Step 1. The subsets ${}^i_pR$ denote the sets of rules with the same left-hand side. Therefore, the cardinality of such set is the number of rules with the same left-hand side (the degree of non-determinism).

Steps 2-3. The set of rules, R_H, is transformed into the new set of rules R' that contains at most two rules with the same left-hand side.

Step 4. The new set of rules, R_D, is initialized by all already deterministic rules from R'.

Step 5. Every pair of rules with the same left-hand side is simulated by the sequence of seven new rules in R_D: (1) generates the auxiliary $(i+1)$-th bounder, (2 and 3) do the selection from one of these simulated rules, (4 and 5) rewrite the ith bounder, (6 and 7) change the state to the target state and erase the auxiliary bounder.

The rigorous version of the proof is left to the kind reader. □

Example 2. Let us show the construction based on the proof of Theorem 1 by transforming CF#RS from Example 1, H, into deterministic version of CF#RS, D. Because the degree of determinism is less than three we can skip first three steps of the construction. Then, we copy all deterministic rules into R_D by actions in step 4, so $R_D = \{1\colon s \, {}_1\# \rightarrow p \, \#\#, 3\colon q \, {}_2\# \rightarrow p \, \#c, 5\colon f \, {}_1\# \rightarrow f \; c\}$ and $Q_D = \{s, p, q, f\}$. Finally, by step 5, we generate the simulating sequence for the pair of rules, $2\colon p \, {}_1\# \rightarrow q \; a\#b$ and $4\colon p \, {}_1\# \rightarrow f \; ab$.
Since $r_i = 2$, $r_j = 4$, $p = p$, $i = 1$, $q_1 = q$, $q_2 = f$, $x_1 = a\#b$, $x_2 = ab$, add the following rules into R_D:

$p \, {}_1\# \rightarrow \langle 2,4 \rangle \, \#\#$
$\langle 2,4 \rangle \, {}_1\# \rightarrow \langle 2 \rangle \, \#$
$\langle 2,4 \rangle \, {}_2\# \rightarrow \langle 4 \rangle \, \#$
$\langle 2 \rangle \, {}_1\# \rightarrow \langle 2' \rangle \; a\#b$
$\langle 4 \rangle \, {}_1\# \rightarrow \langle 4' \rangle \; ab$
$\langle 2' \rangle \, {}_2\# \rightarrow q \, \varepsilon$
$\langle 4' \rangle \, {}_2\# \rightarrow f \, \varepsilon$

At the end, $Q_D = \{s, p, q, f, \langle 2,4 \rangle, \langle 2 \rangle, \langle 4 \rangle, \langle 2' \rangle, \langle 4' \rangle\}$.

□

Corollary 2. *For every $k \geq 1$, $\mathcal{L}_k(CF\#RS) \subseteq \mathcal{L}_{k+1}(detCF\#RS)$.*
Proof. Since there is always only one more bounder added in the construction proof of Theorem 1, the corollary holds.

Theorem 3. For $k = 1$, $card(\mathcal{L}_k(detCF\#RS)) \leq 1$.
Proof. Let assume $k = 1$ and $card(\mathcal{L}_k(detCF\#RS)) > 1$. The determinism implies that every rule has unique left-hand side. Thus, the states and their programming do not allow a branching or selection in the set of rules, thereby, in the computation. In detCF#RS, there is no way how to generate two different configurations from the starting configuration, *s#*, by one or more computational steps. So, the assumption that $card(\mathcal{L}_k(CF\#RS)) > 1$ for $k = 1$ is wrong.

5 CONCLUSION

In fact, the way how the determinism is achieved from non-deterministic CF#RS is the replacement of one type of non-determinism by another. From this point of view, the conclusion of this paper says that the determinism has to be defined in more practical way to be useful, for example in syntax analysis.

Open Problems Area. We still do not know if there exists some construction algorithm that does not increase the index of the detCF#RS during the construction from arbitrary CF#RS of index k. In symbols, does for every $k \geq 2$ hold $\mathcal{L}_k(CF\#RS) = \mathcal{L}_k(detCF\#RS)$?

The other question is the definition of some strong determinism with practical usability.

ACKNOWLEDGEMENT

This work was supported by the MŠMT FRVŠ 673/G1/2007 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.