Theoretical Informatics and Applications Will be set by the publisher
Informatique Théorique et Applications

ON CONTEXT-FREE REWRITING WITH A SIMPLE RESTRICTION AND ITS COMPUTATIONAL COMPLETENESS *

TOMÁŠ MASOPUST [1] AND ALEXANDER MEDUNA [1]

Abstract. This paper discusses context-free rewriting systems in which there exist two disjoint finite sets of rules, and a symbol, referred to as a condition of applicability, is attached to each rule in either of these two sets. In one set, a rule with a symbol attached to it is applicable if the attached symbol occurs in the current rewritten string while in the other set, such a rule is applicable if the attached symbol does not occur there. The present paper demonstrates that these rewriting systems are computationally complete. From this main result, the paper derives several consequences and solves several open problems.

1991 Mathematics Subject Classification. 68Q45, 68Q42.

1. INTRODUCTION

The theory of formal languages has introduced a broad variety of modified context-free grammars in order to increase their generative power. Placing certain conditions of applicability upon grammatical rules represents an important type of these modifications. That is, only grammatical rules satisfying prescribed conditions of applicability are applicable during every derivation step of the generative process. As obvious, keeping these conditions as simple as possible always represents a highly appreciated property of this type of modifications. The present paper discusses context-free rewriting systems with conditions of applicability that satisfy this property.

More specifically, this paper discusses the notion of a context-free rewriting system that contains two disjoint finite sets of rules. A condition of applicability consisting of a single symbol is attached to each rule in either of these two sets. In one set, a rule with a

Keywords and phrases: formal languages, context-free grammar, context-free rewriting system, derivation restriction, generative power.

* *Supported by the Czech Ministry of Education under the Research Plan No. MSM 0021630528 and the Czech Grant Agency project No. 201/07/0005.*

[1] Faculty of Information Technology, Brno University of Technology, Božetěchova 2, Brno 61266, Czech Republic, e-mail: masopust@fit.vutbr.cz, meduna@fit.vutbr.cz

symbol attached to it is applicable if the attached symbol occurs in the current rewritten string while in the other set, such a rule is applicable if the attached symbol does not occur there at all. As its main result, the present paper demonstrates that the resulting rewriting systems are computationally complete.

From this main result, the paper derives several consequences and side results in terms of other conditional grammars, recently summarized in [4]. Specifically, it includes random context grammars (see [7]) and their special variants—semi-conditional grammars (see [5]) and simple semi-conditional grammars (see [3])—and concludes that they are computationally complete.

2. PRELIMINARIES

In this paper, we assume that the reader is familiar with the theory of formal languages (see [1, 6]). For an alphabet (finite nonempty set) V, V^* represents the free monoid generated by V. The unit of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$. For $w \in V^*$, $|w|$ denotes the length of w and $alph(w)$ denotes the set of all symbols occurring in w. Let $\mathscr{L}_{\mathrm{RE}}$ denote the family of recursively enumerable languages.

A *context-free grammar* is a quadruple $G = (N, T, P, S)$, where N is a nonterminal alphabet, T is a terminal alphabet such that $N \cap T = \emptyset$, $V = N \cup T$, $S \in N$ is the start symbol, and P is a finite set of rules of the form $A \to x$, where $A \in N$ and $x \in V^*$. For $u, v \in V^*$, uAv directly derives uxv, denoted by $uAv \Rightarrow uxv$, provided that $A \to x \in P$. The language generated by G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$, where $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$. The family of languages generated by context-free grammars is denoted by $\mathscr{L}_{\mathrm{CF}}$.

2.1. UNORDERED SCATTERED CONTEXT GRAMMARS

An *unordered scattered context grammar* is a quintuple $G = (N, T, P, S, R)$, where N is a nonterminal alphabet, T is a terminal alphabet such that $N \cap T = \emptyset$, $V = N \cup T$, $S \in N$ is the start symbol, P is a finite set of rules of the form

$$(A_1, A_2, \ldots, A_n) \to (w_1, w_2, \ldots, w_n),$$

for some $n \geq 1$, where $A_i \in N$, $w_i \in V^*$, for all $i = 1, \ldots, n$, and

$$R \subseteq \{r_i \to s_i : (r_1, \ldots, r_i, \ldots, r_k) \to (s_1, \ldots, s_i, \ldots, s_k) \in P, 1 \leq i \leq k\}$$

is a set of context-free rules that can be skipped in the parallel application of a finite number of context-free rules if their left-hand sides do not appear in the current sentential form. Specifically, a rule $(A_1, A_2, \ldots, A_n) \to (w_1, w_2, \ldots, w_n)$ from P is applied to a string $x = x_1 A_{i_1} x_2 A_{i_2} \ldots x_u A_{i_u} x_{u+1}$, where $x_i \in V^*$, for all $i = 1, \ldots, u+1$, provided that

(1) $(A_{i_1}, A_{i_2}, \ldots, A_{i_u})$ is a permutation of a subsequence of $(A_1, A_2, \ldots, A_n)$, and
(2) if $A_j \in \{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_u}\}$, then $A_j \notin alph(x)$ and $A_j \to w_j \in R$.

This application results in the string $y = x_1 w_{i_1} x_2 w_{i_2} \dots x_u w_{i_u} x_{u+1}$, written as $x \Rightarrow y$. The language generated by G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$, where $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$.

An unordered scattered context grammar $G = (N, T, P, S, R)$ is said to be *2-limited* if

(1) $(A_1, A_2, \dots, A_n) \to (w_1, w_2, \dots, w_n) \in P$ implies $n \leq 2$ and $|w_i| \leq 2$, $i = 1, 2$, and
(2) $n = 1$ implies $A_1 = S$.

The following theorem is proved in [2].

Theorem 2.1. *Every recursively enumerable language is generated by a 2-limited unordered scattered context grammar.*

3. DEFINITIONS

Definition 3.1. A *conditional context-free rewriting system* is a quintuple

$$G = (N, T, P_+, P_-, S),$$

where N is a nonterminal alphabet, T is a terminal alphabet such that $N \cap T = \emptyset$, $S \in N$ is the start symbol, and P_+ and P_- are two finite sets of rules of the form $(A \to x, X)$, where $A \to x$ is a context-free rule and $X \in N$ is a *condition of applicability*, such that the sets $\{A \to x : (A \to x, X) \in P_+\}$ and $\{A \to x : (A \to x, X) \in P_-\}$ are disjoint.

Remark 3.2. Context-free rules can be thought of as rules of the form $(A \to x, A) \in P_+$.

As usual, $V = N \cup T$ denotes the total alphabet.

Definition 3.3. For all $u, v \in V^*$ and $(A \to x, X) \in P_+ \cup P_-$, uAv directly derives uxv in G, symbolically written as $uAv \Rightarrow uxv$, provided that

(1) either $(A \to x, X) \in P_+$ and $X \in alph(uAv)$,
(2) or $(A \to x, X) \in P_-$ and $X \notin alph(uAv)$.

The language generated by G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$, where $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$. The family of languages generated by conditional context-free rewriting systems is denoted by $\mathscr{L}_{\text{CCFRS}}$.

Remark 3.4. For the better readability, we often write $(A \to x, X)_+$ and $(A \to x, X)_-$ instead of $(A \to x, X) \in P_+$ and $(A \to x, X) \in P_-$, respectively.

4. MAIN RESULT

This section demonstrates that $\mathscr{L}_{\text{CCFRS}} = \mathscr{L}_{RE}$. From this main result, several corollaries follow. Some of them represent well-known results proved in an alternative way. Specifically, these results demonstrate that random context grammars (see [7]) and semi-conditional grammars of degree (i, j), for all $i, j \geq 1$, (see [5]) are computationally complete. More importantly, some other corollaries solve open problems in formal language theory. For instance, they establish that simple semi-conditional grammars of degree (i, j), for all $i, j \geq 1$, (see [3]) are computationally complete.

As the proof of $\mathscr{L}_{\mathrm{CCFRS}} = \mathscr{L}_{RE}$ is complicated, we structure this section into several subsections to make the proof as readable as possible.

Theorem 4.1. $\mathscr{L}_{\mathrm{CCFRS}} = \mathscr{L}_{\mathrm{RE}}$.

Before we prove Theorem 4.1, we need to slightly modify 2-limited unordered scattered context grammars in order not to have the start symbol on the right-hand side of any rule.

Lemma 4.2. *Every recursively enumerable language is generated by a 2-limited unordered scattered context grammar $G = (N,T,P,S,R)$, where S does not occur on the right-hand side of any rule from P; i.e., if $(A,B) \to (x,y) \in P$, then $S \notin alph(xy)$, and if $(S) \to (w) \in P$, then $S \notin alph(w)$.*

Proof. Let L be a recursively enumerable language and $G = (N,T,P,S,R)$ be a 2-limited unordered scattered context grammar such that $L = L(G)$. Assume that S occurs on the right-hand side of a rule from P. Construct an unordered scattered context grammar, G', satisfying the lemma as follows.

Let S' and S_1 be two new nonterminals not contained in N, and set $N' = N \cup \{S', S_1\}$, $P' = P \cup \{(S') \to (S_1 S)\}$, and replace all rules of the form $(S) \to (w)$ in P' with $(S_1, S) \to (S_1, w)$ and $(S_1, S) \to (\varepsilon, w)$. Thus, $G' = (N', T, P', S', R)$ and it is not hard to see that $L(G) = L(G')$. □

4.1. CONSTRUCTION

Let L be a recursively enumerable language. Without loss of generality, we can assume that L is generated by a 2-limited unordered scattered context grammar $G = (N,T,P,S,R)$ satisfying Lemma 4.2. Let n be the number of rules in P. Then,

$$P = \bigcup_{i=1}^{u-1} ((S) \to (w_i)) \cup \bigcup_{i=u}^{n} ((A_{i1}, A_{i2}) \to (w_{i1}, w_{i2})),$$

where $1 \leq u \leq n$. We construct a conditional context-free rewriting system

$$G' = (N', T, P_+, P_-, S')$$

as follows:

- For each $A \in N$, let A' and A'' be new nonterminals.
- Let #, \$, X, X', X'', X''', Y, Z be new nonterminals.

Then, set

$$N' = N \cup \{A' : A \in N\} \cup \{A'' : A \in N\} \cup \{\#, \$, X, X', X'', X''', Y, Z\} \cup \{[p]_i : p \in P, 0 \leq i \leq 7\},$$

$V = N \cup T$, and $V' = N' \cup T$. Finally, define P_+ and P_- as follows:

(1) for each $(S) \to (w) \in P$, add $(S \to w, S)$ to P_+.

(2) for each $p = (A,B) \to (x,y) \in P$, add the following rules to $P_+ \cup P_-$:
  (a) $(A \to X'\#A', \#)_-$
  (b) $(X' \to X'', Y)_-$
  (c) $(X'' \to X, Z)_-$
  (d) $(B \to X'''\$B', \$)_-$
  (e) $(X''' \to \varepsilon, Z)_-$
  (f) $(X \to [p]_0, A')_+$
  (g) $(A' \to A'', [p]_0)_+$
  (h) $([p]_0 \to [p]_1, A'')_+$
  (i) $([p]_1 \to [p]_2, B')_+$
  (j) $([p]_2 \to [p]_3, X''')_-$
  (k) $(A'' \to x, [p]_3)_+$
  (l) $(B' \to y, [p]_3)_+$
  (m) $([p]_3 \to [p]_4, A'')_-$
  (n) $([p]_4 \to YZ, B')_-$
  (o) $(\# \to \varepsilon, Y)_+$
  (p) $(\$ \to \varepsilon, Z)_+$
  (q) $(Y \to \varepsilon, X')_-$
  (r) $(Z \to \varepsilon, X''')_-$

(3) If $B \to y \in R$, add to $P_+ \cup P_-$ also
  (s) $([p]_1 \to [p]_5, B)_-$
  (t) $([p]_5 \to [p]_6, \$)_-$
  (u) $([p]_6 \to [p]_7, B'')_-$
  (v) $(A'' \to x, [p]_7)_+$
  (w) $([p]_7 \to Y, A'')_-$

(4) If $A \to x \in R$, add $(B,A) \to (y,x)$ to P.

4.2. BASIC IDEA

Clearly, the basic idea of the proof is to show how G' simulates G, and, on the other hand, to show that every successful derivation of G' can be simulated by G. To give an insight into the proof, we first explain the meaning of the newly introduced symbols:

- Symbols A' and A'' corresponding to each $A \in N$ are introduced to mark the symbols we want to replace during the simulation of one derivation step of G.
- Symbols # and \$ are introduced to block the simulation of another rule; of course, if there is # or \$ in the sentential form, we cannot mark any other pair of nonterminals (see rules (a) and (d)).
- Symbols X, X', X'', and X''' are introduced to make sure that the previous simulation of a rule is successfully finished.
- All the symbols $[p]_i$ check (step by step) that there are two marked symbols, say A and B, in the sentential form corresponding to two symbols of the left-hand side of $p = (A,B) \to (x,y)$ (or only one symbol, A, if there is no symbol B in the sentential form and $B \to y \in R$).

- Finally, by symbols Y and Z, the system verifies that the simulation of p is successfully completed.

Now, we briefly explain how the rewriting system is synchronized. Let $p = (A,B) \to (x,y) \in P$. Then, the rewriting system works as follows.

- Rules (a) and (d) mark symbols A and B and introduce X', X''', #, and \$ in the current sentential form. As mentioned above, while there are # and \$ in the sentential form, the system cannot start the simulation of another rule. Then, by using X', X'', and X''', the system makes sure that there is no Y or Z in the current sentential form; i.e., the previous simulation is finished.
- Then, rule (f) nondeterministically selects a rule of G, say for instance p, of the form $(A,B) \to (x,y)$ to simulate.
- Rules (g) through (m) then simulate the application of p so that they first check that there are marked symbols A and B, and if so, A and B are replaced with x and y, respectively.
- By rule (n), the system verifies that the simulation has been successful by introducing symbols Y and Z. Notice that there are still # and \$ in the current sentential form. Therefore, the simulation of another rule is not possible.
- Finally, rules (o) and (p) remove # and \$, respectively, and then rules (q) and (r) remove Y and Z, respectively. After eliminating # or \$, the simulation of another rule begins. However, symbols Y and Z have to be removed before X (allowing a new nondeterministic selection of a rule to be simulated) is introduced in the current sentential form (see rules (b) and (c)).

Note that one could see this synchronization as a semaphore, where

(1) some processes (rules of G to be applied) want to reach the critical section trying so by rules (a) through (e) nondeterministically marking two nonterminals to be rewritten inside of the critical section,
(2) a process reaches the critical section, where it simulates the derivation of a rule of G replacing the marked symbols and introduces Y and Z if the simulation is successful (see rules (f) through (n)), and, finally,
(3) the process leaves the critical section by removing symbols #, \$, Y, and Z, and allows another process to enter the critical section (see rules (o) through (r)).

4.3. PROOF OF $L(G) \subseteq L(G')$

To prove that $L(G) \subseteq L(G')$, consider a derivation step in G, $w_1Aw_2Bw_3 \Rightarrow w_1xw_2yw_3$, according to $p = (A,B) \to (x,y)$. Then, in G', this derivation step is simulated as follows. Note that in what follows, $\Rightarrow_{(x)}$ means that the derivation step $\Rightarrow$ is made by an application of rule (x).

$$
\begin{array}{llll}
w_1Aw_2Bw_3 & \Rightarrow_{(a)} & w_1X'\#A'w_2Bw_3 & \Rightarrow_{(d)} \quad w_1X'\#A'w_2X'''\$B'w_3 \\
& \Rightarrow_{(b)} & w_1X''\#A'w_2X'''\$B'w_3 & \Rightarrow_{(c)} \quad w_1X\#A'w_2X'''\$B'w_3 \\
& \Rightarrow_{(e)} & w_1X\#A'w_2\$B'w_3 & \Rightarrow_{(f)} \quad w_1[p]_0\#A'w_2\$B'w_3 \\
& \Rightarrow_{(g)} & w_1[p]_0\#A''w_2\$B'w_3 & \Rightarrow_{(h)} \quad w_1[p]_1\#A''w_2\$B'w_3 \\
& \Rightarrow_{(i)} & w_1[p]_2\#A''w_2\$B'w_3 & \Rightarrow_{(j)} \quad w_1[p]_3\#A''w_2\$B'w_3
\end{array}
$$

$$\begin{array}{llll} \Rightarrow_{(k)} & w_1[p]_3\#xw_2\$B'w_3 & \Rightarrow_{(l)} & w_1[p]_3\#xw_2\$yw_3 \\ \Rightarrow_{(m)} & w_1[p]_4\#xw_2\$yw_3 & \Rightarrow_{(n)} & w_1YZ\#xw_2\$yw_3 \\ \Rightarrow_{(o)} & w_1YZxw_2\$yw_3 & \Rightarrow_{(p)} & w_1YZxw_2yw_3 \\ \Rightarrow_{(q)} & w_1Zxw_2yw_3 & \Rightarrow_{(r)} & w_1xw_2yw_3\,. \end{array}$$

If there is no B in the sentential form and $B \to y \in R$, the derivation step is simulated as follows.

$$\begin{array}{llll} w_1Aw_2 & \Rightarrow_{(a)} & w_1X'\#A'w_2 & \Rightarrow_{(b)} & w_1X''\#A'w_2 \\ & \Rightarrow_{(c)} & w_1X\#A'w_2 & \Rightarrow_{(f)} & w_1[p]_0\#A'w_2 \\ & \Rightarrow_{(g)} & w_1[p]_0\#A''w_2 & \Rightarrow_{(h)} & w_1[p]_1\#A''w_2 \\ & \Rightarrow_{(s)} & w_1[p]_5\#A''w_2 & \Rightarrow_{(t)} & w_1[p]_6\#A''w_2 \\ & \Rightarrow_{(u)} & w_1[p]_7\#A''w_2 & \Rightarrow_{(v)} & w_1[p]_7\#xw_2 \\ & \Rightarrow_{(w)} & w_1Y\#xw_2 & \Rightarrow_{(o)} & w_1Yxw_2 \\ & \Rightarrow_{(q)} & w_1xw_2\,. & & \end{array}$$

The proof then proceeds by induction. □

4.4. PROOF OF $L(G') \subseteq L(G)$

To prove that $L(G') \subseteq L(G)$, consider a successful derivation of G'. Since the applicability of rules depends only on occurrences of symbols in sentential forms, we disregard their positions. That is, such a successful derivation begins by a derivation of the form $S \Rightarrow^* w$, where $w \in \{w_1Aw_2Bw_3, w_1Aw_2 : w_1w_2w_3 \in V^*, A, B \in N\}$. In the rest of this proof, we proceed as follows:

- **A:** We explore all possible beginnings of successful derivations starting from w.
- **B:** We demonstrate that all the beginnings of derivations in **A** simulate the application of a rule of the form $(A,B) \to (x,y)$.

Let w be of the form $w_1Aw_2Bw_3$. Figure 1 presents all possible successful derivations starting from w. Now, we explain why some rules are not applicable in a successful derivation so that we explore all possible derivations starting from the sentential forms from nodes of the graph in Figure 1.

Clearly, in node 0, i.e., in the sentential form $w_1Aw_2Bw_3$, only rules constructed in (a) and (d) are applicable. If (a) is applied, # is introduced to the sentential form, and while there is # in the sentential form, no other rule constructed in (a) can be applied. Thus, only rules constructed in (b) and (d) are applicable in node 1. Analogously for the other nodes. Note that in some cases, there are applicable rules that are not depicted in Figure 1. We now show that such rules block the derivation, i.e., the derivation is not able to derive a terminal string. These rules are listed below with respect to numbers of nodes in which they are applicable.

- **Node 16:** Only rules constructed in (g) (in case $A = B$), (h), and (e) are applicable. However, if (g) is applied, the derivation cannot generate a terminal string; indeed, Y cannot be generated because neither $[p]_2$ nor $[p]_6$ can be generated.
- **Node 18:** Only rules constructed in (i), (e), and (s) are applicable. However, if rule (s) is applied, then only (e) is applicable, i.e., $w_1[p]_1\#A''w_2X'''\$B'w_3 \Rightarrow_{(s)} w_1[p]_5\#A''w_2X'''\$B'w_3 \Rightarrow_{(e)} w_1[p]_5\#A''w_2\$B'w_3$, and the derivation is blocked.

Node 19: Only rules (g) and (h) are applicable. However, applying (g), the derivation cannot generate a terminal string as mentioned above (see Node 16).

Node 21: Only rules constructed in (i) and (s) are applicable. However, if (s) is applied, we get the following sentential form, $w_1[p]_5\#A''w_2\$B'w_3$, and the derivation is blocked.

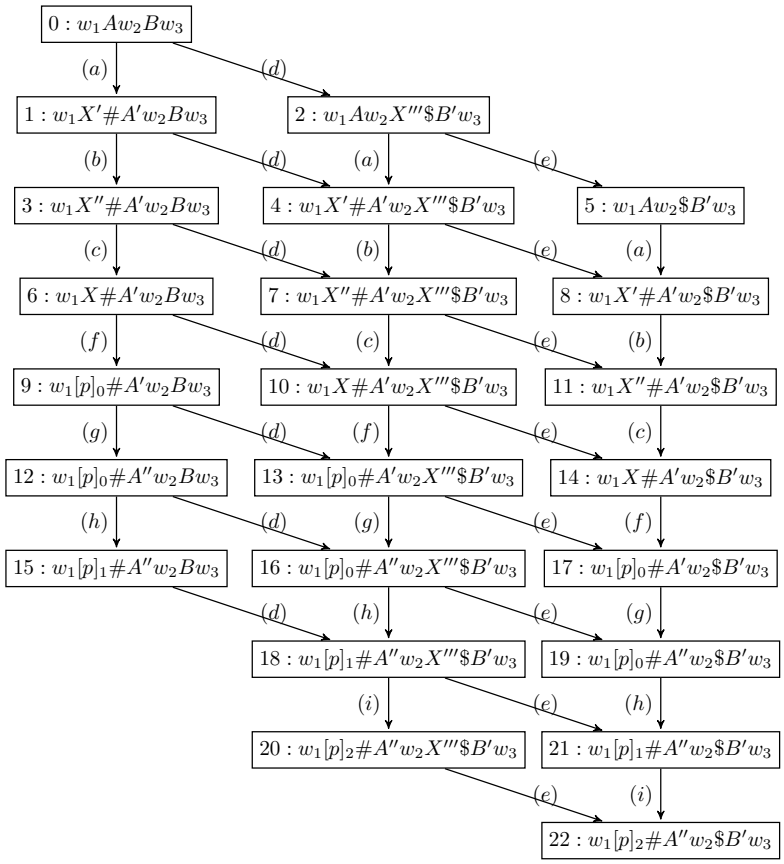

FIGURE 1. All possible beginnings of successful derivations starting from $w_1Aw_2Bw_3$. (Nodes are numbered; in this case from 0 to 22.)

In all cases, we have generated the string $w_1[p]_2\#A''w_2\$B'w_3$. The derivation now continues as shown in Figure 2.

Node 24: Only rules constructed in (m) and (l) are applicable. However, if rule (m) is applied, i.e., we get $w_1[p]_4\#xw_2\$B'w_3$, the derivation is blocked.

Thus, we have derived the following sentential form $w_1YZ\#xw_2\$yw_3$. The derivation now continues as shown in Figure 3.

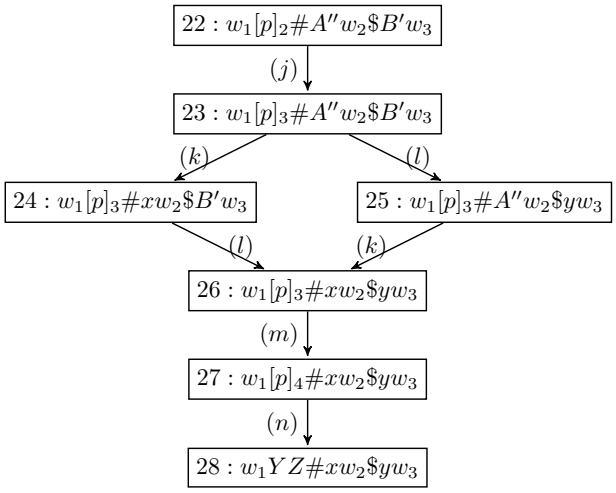

FIGURE 2. All possible beginnings of successful derivations starting from $w_1[p]_2\#A''w_2\$B'w_3$.

Node 28: Only rules (o), (p), (q), and (r) are applicable. If (q) is applied, i.e., $w_1YZ\#xw_2\$yw_3 \Rightarrow_{(q)} w_1Z\#xw_2\$yw_3$, then only rules (r) and (p) are applicable. If (r) is applied, the sentential form is $w_1\#xw_2\$yw_3$, and the derivation is blocked. If (p) is applied, the sentential form is $w_1Z\#xw_2yw_3$, and the derivation is blocked as in (2) below.

If (r) is applied, $w_1YZ\#xw_2\$yw_3 \Rightarrow_{(r)} w_1Y\#xw_2\$yw_3$, then only rules (q) and (o) are applicable. If (q) is applied, the derivation is blocked. If (o) is applied, the sentential form is $w_1Yxw_2\$yw_3$, and the derivation is blocked as in (1) below.

Node 29: Only rules (a), (p), (q), and (r) are applicable. If rule (a) is applied, we get $v_1X'\#C'v_2YZxw_2\$yw_3$, and the derivation is blocked according to rules (b) and (q) because neither Y nor X' can be removed.

If (r) is applied, i.e.,

$$w_1YZxw_2\$yw_3 \quad \Rightarrow_{(r)} \quad w_1Yxw_2\$yw_3\,, \tag{1}$$

then only rules (a) and (q) are applicable. If (a) is applied, Y and X' are in the sentential form—the derivation is blocked. If (q) is applied, the sentential form is $w_1xw_2\$yw_3$, and the derivation is blocked as in (6) below.

Node 30: Only rules (d), (o), (q), and (r) are applicable. If (d) is applied, the current sentential form contains Z and X'''—the derivation is blocked.

If (q) is applied, i.e.,

$$w_1YZ\#xw_2yw_3 \quad \Rightarrow_{(q)} \quad w_1Z\#xw_2yw_3\,, \tag{2}$$

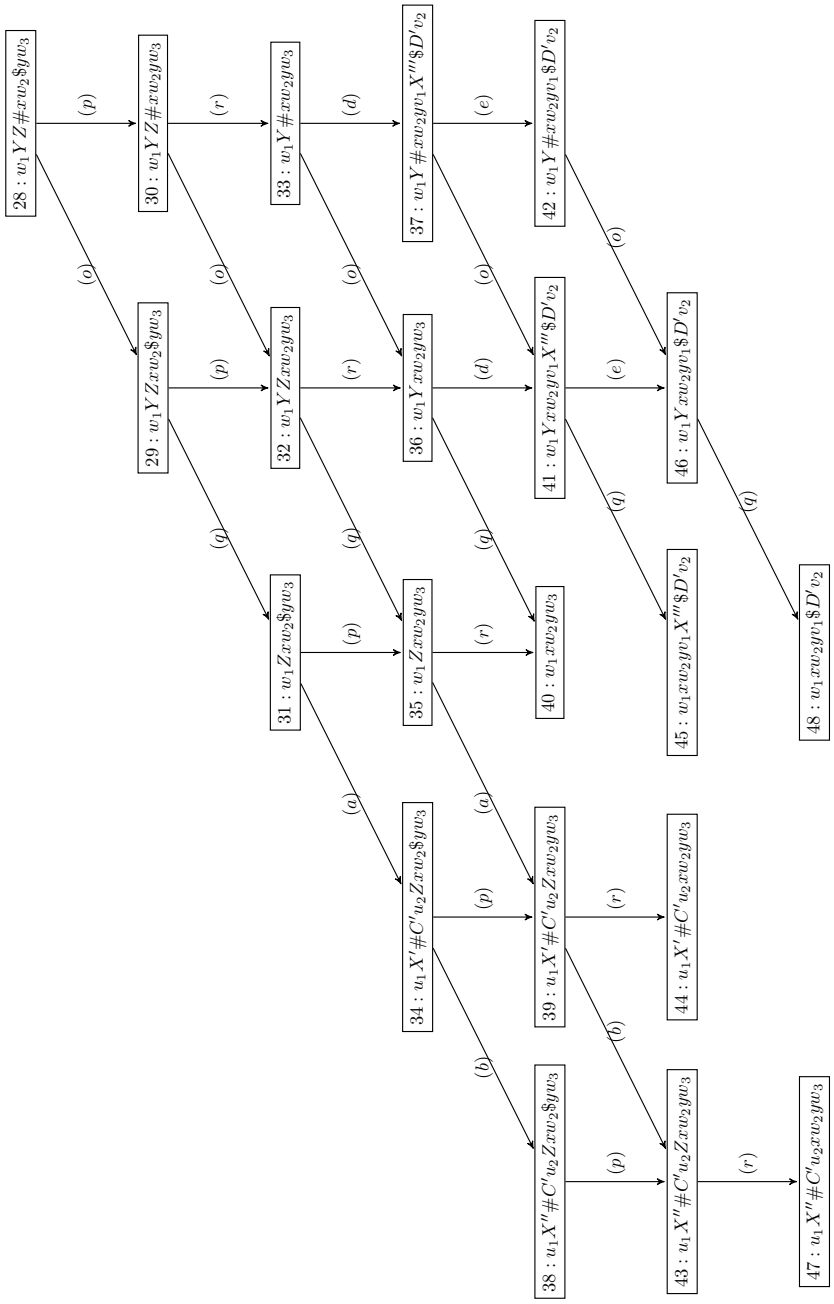

FIGURE 3. All possible beginnings of successful derivations starting from $w_1YZ\#xw_2\$yw_3$.

the only (d) and (r) are applicable. If (d) is applied, the derivation is blocked; Z and X''' are in the sentential form. If (r) is applied, then only rule (d) followed by (e) is applicable, i.e.,

$$\begin{aligned} w_1Z\#xw_2yw_3 &\Rightarrow_{(r)} w_1\#xw_2yw_3 && (3)\\ &\Rightarrow_{(d)} w_1\#xw_2yv_1X'''\$D'v_2 && (4)\\ &\Rightarrow_{(e)} w_1\#xw_2yv_1\$D'v_2\,. && (5)\end{aligned}$$

Again, the derivation is blocked.

Node 31: Only rules (a), (p), and (r) are applicable. If (r) is applied, then only the following sequence of rules is applicable: (a), (b), (c), (f), (g), (h), and (s), i.e.,

$$\begin{aligned} w_1Zxw_2\$yw_3 &\Rightarrow_{(r)} w_1xw_2\$yw_3 && (6)\\ &\Rightarrow_{(a)} u_1X'\#C'u_2xw_2\$yw_3 && (7)\\ &\Rightarrow_{(b)} u_1X''\#C'u_2xw_2\$yw_3 && (8)\\ &\Rightarrow_{(c)} u_1X\#C'u_2xw_2\$yw_3 && (9)\\ &\Rightarrow_{(f)} u_1[q]_0\#C'u_2xw_2\$yw_3 && (10)\\ &\Rightarrow_{(g)} u_1[q]_0\#C''u_2xw_2\$yw_3 && (11)\\ &\Rightarrow_{(h)} u_1[q]_1\#C''u_2xw_2\$yw_3 && (12)\\ &\Rightarrow_{(s)} u_1[q]_5\#C''u_2xw_2\$yw_3\,, && (13)\end{aligned}$$

and the derivation is blocked.

Node 32: Only rules (a), (d), (q), and (r) are applicable. If (a) is applied, the sentential form contains Y and X'; the derivation is blocked. If (d) is applied, we get $w_1YZxw_2yv_1X'''\$D'v_2$, and the derivation is blocked according to rules (e) and (r).

Node 33: Only (d), (q), and (o) are applicable. If (q) is applied, we get $w_1\#xw_2yw_3$, and the derivation is blocked as in (3).

Node 34: Only rules (b), (p), and (r) are applicable. If (r) is applied, the sentential form is $u_1X'\#C'u_2xw_2\$yw_3$, and the derivation is blocked as in (7).

Node 35: Only (a), (d), and (r) are applicable. However, if (d) is applied, we get $w_1Zxw_2yv_1X'''\$D'v_2$, and the derivation is blocked, see rules (e) and (r).

Node 36: Only (a), (d), and (q) are applicable. If (a) is applied, the sentential form is $u_1X'\#C'u_2Yxw_2yw_3$, and the derivation is blocked, see rules (b) and (q).

Node 37: Only (q), (o), and (e) are applicable. If (q) is applied, we get the sentential form $w_1\#xw_2yv_1X'''\$D'v_2$, and the derivation is blocked as in (4).

Node 38: Only (p) and (r) are applicable. If rule (r) is applied, the sentential form is $u_1X''\#C'u_2xw_2\$yw_3$, and the derivation is blocked as in (8).

Node 39: Only (b), (d), and (r) are applicable. However, if (d) is applied, we get $u_1X'\#C'u_2Zxw_2yv_1X'''\$D'v_2$, and the derivation is blocked, see (e) and (r).

Node 41: Only (a), (q), and (e) are applicable. If (a) is applied, the sentential form is $u_1X'\#C'u_2Yxw_2yv_1X'''\$D'v_2$, and the derivation is blocked, see rules (b) and (q).

Node 42: Only (q) and (o) are applicable. If rule (q) is applied, the sentential form is $w_1\#xw_2yv_1\$D'v_2$, and the derivation is blocked as in (5).

Node 43: Only (d) and (r) are applicable. If (d) is applied, the sentential form is $u_1X''\#C'u_2Zxw_2yv_1X'''\$D'v_2$, and the derivation is blocked.

Node 46: Only (a) and (q) are applicable. However, if (a) is applied, the derivation is blocked; Y and X' are in the sentential form.

Thus, $S \Rightarrow^* w \Rightarrow^* \alpha \Rightarrow^* z$, where $z \in T^*$ and α is in one of the following sentential forms:

(i) $w_1xw_2yw_3$
(ii) $u_1X'\#C'u_2xw_2yw_3$
(iii) $u_1X''\#C'u_2xw_2yw_3$
(iv) $w_1xw_2yv_1X'''\$D'v_2$
(v) $w_1xw_2yv_1\$D'v_2$

In this moment, we have simulated rule $p = (A,B) \to (x,y)$, and the derivation continues starting to simulate another rule, $q = (C,D) \to (u,v)$. The rule $(A,B) \to (x,y)$ does the same in G. The proof now proceeds by induction from nodes 0, 1, 3, 2, 5, respectively, or as follows if there is no D (D' and D'') in the sentential form.

Above, we have considered the sentential form w containing B. Now, assume that there is no B in w. Then, the following sequence of rules has to be applied in a successful derivation: (a), (b), (c), (f), (g), and (h);

$$\begin{array}{lllllll} w_1Aw_2 & \Rightarrow_{(a)} & w_1X'\#A'w_2 & \Rightarrow_{(b)} & w_1X''\#A'w_2 & \Rightarrow_{(c)} & w_1X\#A'w_2 \\ & \Rightarrow_{(f)} & w_1[p]_0\#A'w_2 & \Rightarrow_{(g)} & w_1[p]_0\#A''w_2 & \Rightarrow_{(h)} & w_1[p]_1\#A''w_2\,. \end{array}$$

Clearly, as there are no B, B', and B'' (if $B = A$) in the current sentential form, rules (s), (t), and (u), respectively, can be applied.

$$\begin{array}{lllllllll} w_1[p]_1\#A''w_2 & \Rightarrow_{(s)} & w_1[p]_5\#A''w_2 & \Rightarrow_{(t)} & w_1[p]_6\#A''w_2 & \Rightarrow_{(u)} & w_1[p]_7\#A''w_2 \\ & \Rightarrow_{(v)} & w_1[p]_7\#xw_2 & \Rightarrow_{(w)} & w_1Y\#xw_2\,. \end{array}$$

Last two derivation steps follow from the construction. Now, the derivation continues as in Node 33. The rule $(A,B) \to (x,y)$ with $B \to y \in R$ does the same in G. □

4.5. COROLLARIES

Corollary 4.3. *The following families of languages coincide.*

- *The family of languages generated by conditional context-free rewriting systems.*
- *The family of recursively enumerable languages.*
- *The family of languages generated by random context grammars.*
- *The family of languages generated by semi-conditional grammars of degree* $(1,1)$.
- *The family of languages generated by simple semi-conditional grammars of degree* $(1,1)$.

Proof. Conditional context-free rewriting systems are special variants of random context grammars, semi-conditional grammars, and simple semi-conditional grammars. □

The authors thank the anonymous referee for valuable comments and suggestions.

REFERENCES

[1] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.
[2] O. Mayer. Some restrictive devices for context-free grammars. *Information and Control*, 20:69–92, 1972.
[3] A. Meduna and A. Gopalaratnam. On semi-conditional grammars with productions having either forbidding or permitting conditions. *Acta Cybernetica*, 11(4):307–324, 1994.
[4] A. Meduna and M. Švec. *Grammars with Context Conditions and Their Applications*. John Wiley & Sons, New York, 2005.
[5] Gh. Păun. A variant of random context grammars: Semi-conditional grammars. *Theoretical Computer Science*, 41:1–17, 1985.
[6] A. Salomaa. *Formal languages*. Academic Press, New York, 1973.
[7] A. P. J. van der Walt. Random context grammars. In *Proceedings of the Symposium on Formal Languages*. 1970.

Communicated by (The editor will be set by the publisher).
(The dates will be set by the publisher).