

# Two Results on Discontinuous Input Processing

Vojtěch Vorel<sup>1\*</sup>

Faculty of Mathematics and Physics, Charles University, Malostranské nám. 25,  
Prague, Czech Republic,  
vorel@ktiml.mff.cuni.cz

**Abstract.** First, we show that universality and other properties of general jumping finite automata are undecidable, which answers a question asked by Meduna and Zemek in 2012. Second, we close the study raised by Černo and Mráz in 2010 by proving that clearing restarting automata using contexts of size two can accept binary non-context-free languages.

## 1 Introduction

In 2012, Meduna and Zemek [8,9] introduced *general jumping finite automata* as a model of discontinuous information processing. A general jumping finite automaton (GJFA) is described by a finite set  $Q$  of states, a finite alphabet  $\Sigma$ , a finite set  $R$  of *rules* from  $Q \times \Sigma^* \times Q$ , an initial state  $q_0 \in Q$ , and a set  $F \subseteq Q$  of final states. In a step of computation, the automaton switches from a state  $r$  to a state  $s$  using a rule  $(r, v, s) \in R$ , and deletes a factor equal to  $v$  from any part of the input word. The choices of the rule used and of the factor deleted are made nondeterministically (in other words, the read head can jump to any position). A word is accepted if there is a computation resulting in the empty word. The boldface term **GJFA** refers to the class of languages accepted by GJFA. The initial work [8,9] deals mainly with closure properties of **GJFA** and its relations to classical language classes (the publications contain flaws, see [13]). It turns out that the class **GJFA** does not have Boolean closure properties (complementation, intersection) nor closure properties related to continuous processing (concatenation, Kleene star, homomorphism, inverse homomorphism, shuffle). Accordingly, the class also does not stick to classical complexity measures - it is incomparable with both regular and context-free languages. It is a proper subclass of both context-sensitive languages and of the class NP, while there exist NP-complete **GJFA** languages. See [2], which is an extended version of [3].

On the other hand, the concept of *restarting automata* [7,10] is motivated by reduction analysis and grammar checking of natural language sentences. In 2010, Černo and Mráz [12] introduced a subclass named *clearing restarting automata* (cl-RA) in order to describe systems that use only very basic types of reduction

---

\* Research supported by the Czech Science Foundation grant GA14-10799S and the GAUK grant No. 52215.

rules (see also [11]). Unlike GJFA, clearing restarting automata may delete factors according to contexts and endmarks, but they are not controlled by state transitions. A key property of each cl-RA is the maximum length of context used. For  $k \geq 1$ , a *k-clearing restarting automaton* (*k-cl-RA*) is described by a finite alphabet  $\Sigma$  and a finite set  $I$  of instructions of the form  $(u_L, v, u_R)$ , where  $v \in \Sigma^*$ ,  $u_L \in \Sigma^k \cup \S \Sigma^{k-1}$ , and  $u_R \in \Sigma^k \cup \Sigma^{k-1} \$$ . The words  $u_L, u_R$  specify left and right context for consuming a factor  $v$ , while  $\S$  and  $\$$  stand for the left and right end of input, respectively.

## 2 Preliminaries

We heavily use the notion of *insertion*, as it was described, e.g., in [1,4,6]:

**Definition 1.** Let  $K, L \subseteq \Sigma^*$  be languages. The insertion of  $K$  to  $L$  is

$$L \leftarrow K = \{u_1 v u_2 \mid u_1 u_2 \in L, v \in K\}.$$

More generally, for each  $k \geq 1$  we denote

$$\begin{aligned} L \leftarrow^k K &= (L \leftarrow^{k-1} K) \leftarrow K, \\ L \leftarrow^* K &= \bigcup_{i \geq 0} L \leftarrow^i K, \end{aligned}$$

where  $L \leftarrow^0 K$  stands for  $L$ . In expressions with  $\leftarrow$  and  $\leftarrow^*$ , a singleton set  $\{w\}$  may be replaced by  $w$ .

A chain  $L_1 \leftarrow L_2 \leftarrow \dots \leftarrow L_d$  of insertions is evaluated from the left, e.g.  $L_1 \leftarrow L_2 \leftarrow L_3$  means  $(L_1 \leftarrow L_2) \leftarrow L_3$ . Finally,  $L \subseteq \Sigma^*$  is a unitary language if  $L = w \leftarrow^* K$  for  $w \in \Sigma^*$  and finite  $K \subseteq \Sigma$ .

As described above, a GJFA is a quintuple  $M = (Q, \Sigma, R, q_0, F)$ . The original definition of the accepted language  $L(M)$  is based on *configurations* that specify a position of the read head (i.e., starting positions of the factor to be erased in the next step). For our proofs, this type of configurations is useless, whence we save space by directly using the following generative characterization [13, Corollary 1] of  $L(M)$  as a definition:

**Definition 2.** Let  $M = (Q, \Sigma, R, s, F)$  be a GJFA and  $w \in \Sigma^*$ . Then  $w \in L(M)$  if and only if  $w = \epsilon$  and  $s \in F$ , or

$$w \in \epsilon \leftarrow v_d \leftarrow v_{d-1} \leftarrow \dots \leftarrow v_2 \leftarrow v_1, \quad (1)$$

where  $d \geq 1$  and  $v_1, v_2, \dots, v_d$  is a labeling of an accepting path in  $M$ .

**Definition 3.** For  $k \geq 0$ , a *k-context rewriting system* is a tuple  $R = (\Sigma, \Gamma, I)$ , where  $\Sigma$  is an input alphabet,  $\Gamma \supseteq \Sigma$  is a working alphabet not containing the special symbols  $\S$  and  $\$$ , called sentinels, and  $I$  is a finite set of instructions of the form

$$(u_L, v \rightarrow t, u_R),$$

where  $u_L$  is a left context,  $x \in \Gamma^k \cup \epsilon \Gamma^{k-1}$ ,  $y$  is a right context,  $y \in \Gamma^k \cup \Gamma^{k-1} \epsilon$ , and  $v \rightarrow t$  is a rule,  $z, t \in \Gamma^*$ . A word  $w = u_1 v u_2$  can be rewritten into  $u_1 t u_2$  (denoted  $asu_1 v u_2 \rightarrow_R u_1 t u_2$ ) if and only if there exists an instruction  $(u_L, v \rightarrow t, u_R) \in I$  such that  $u_L$  is a suffix of  $\epsilon u_1$  and  $u_R$  is a prefix of  $u_2 \epsilon$ . The symbol  $\rightarrow_R^*$  denotes the reflexive-transitive closure of  $\rightarrow_R$ .

**Definition 4.** For  $k \geq 0$ , a  $k$ -clearing restarting automaton ( $k$ -cl-RA) is a system  $M = (\Sigma, I)$ , where  $(\Sigma, \Sigma, I)$  is a  $k$ -context rewriting system such that for each  $\mathbf{i} = (u_L, v \rightarrow t, u_R) \in I$  it holds that  $v \in \Sigma^+$  and  $t = \epsilon$ . Since  $t$  is always the empty word, we use the notation  $\mathbf{i} = (u_L, v, u_R)$ . A  $k$ -cl-RA  $M$  accepts the language

$$L(M) = \{w \in \Sigma^* \mid w \vdash_M^* \epsilon\},$$

where  $\vdash_M$  denotes the rewriting relation  $\rightarrow_M^*$  of  $\overline{M} = (\Sigma, \Sigma, I)$ . The term  $\mathcal{L}(k\text{-cl-RA})$  denotes the class of languages accepted by  $k$ -cl-RA.

Like in GJFA, one may consider the generative approach to languages accepted by clearing restarting automata. In this case, the generative approach is formalized by writing  $w_2 \dashv w_1$  instead of  $w_1 \vdash w_2$ .

### 3 Undecidability in General Jumping Finite Automata

This section proves the following theorem, which solves an open problem stated in [9,8]:

**Theorem 5.** Given a GJFA  $M = (Q, \Sigma, R, s, F)$ , it is undecidable whether  $L(M) = \Sigma^*$ .

*Proof.* Given a context-free grammar  $G$  with terminal alphabet  $\Sigma_T$ , it is undecidable whether  $L(G) = \Sigma_T^*$  [5]. We present a reduction from this problem to the universality of GJFA. Assume that the given grammar  $G$ :

- has non-terminal alphabet  $\Sigma_N = \{A_1, \dots, A_m\}$  with a start symbol  $A_S \in \Sigma_N$ ,
- does not accept the empty word, and
- is given in Greibach normal form [5] as

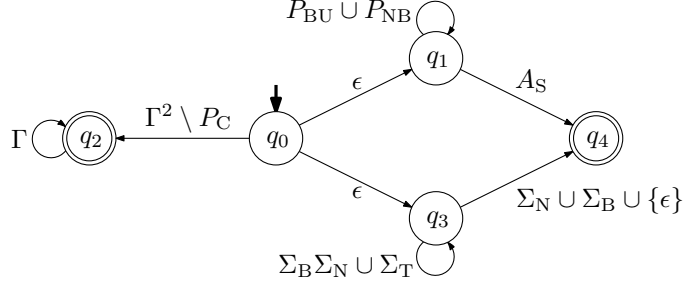
$$B_i \rightarrow u_i,$$

where  $B_i \in \Sigma_N$  and  $u_i \in \Sigma_T \Sigma_N^*$  for  $i \in \{1, \dots, n\}$ .

We construct a GJFA  $M_G = (Q, \Gamma, R, s, F)$  as follows, denoting  $\Sigma_B = \{b_1, \dots, b_m\}$ :

$$\begin{aligned} Q &= \{q_0, q_1, q_2, q_3, q_4\}, \\ \Gamma &= \Sigma_T \cup \Sigma_N \cup \Sigma_B, \end{aligned}$$

$s = q_0$ ,  $F = \{q_4\}$ , and  $R$  follows Figure 1. Each arrow labeled with a finite set

**Fig. 1.** The GJFA  $M_G$ 

$S \subseteq \Gamma^*$  stands for  $|S|$  transitions, each labeled by a unique  $v \in S$ . The following finite sets are used:

$$\begin{aligned}
 P_{BU} &= \{b_i u_i \mid i = 1, \dots, m\}, \\
 P_{NB} &= \{A_i b_i \mid i = 1, \dots, m\}, \\
 P_C &= \{x A_1 \mid x \in \Sigma_T\} \cup \\
 &\quad \cup \{A_i b_i \mid i = 1, \dots, m\} \cup \\
 &\quad \cup \{b_i A_{i+1} \mid i = 1, \dots, m-1\} \cup \\
 &\quad \cup \{b_m x \mid x \in \Sigma_T\}.
 \end{aligned}$$

For a word  $w \in \Gamma^*$  we denote with  $w_T$  and  $w_{N,B}$  the projections of  $w$  to sub-alphabets  $\Sigma_T$  and  $\Sigma_N \cup \Sigma_B$  respectively.<sup>1</sup> Let us show that  $L(G) = \Sigma_T^*$  if and only if  $L(M_G) = \Gamma^*$ .

First, suppose that  $L(G) = \Sigma_T^*$  and take an arbitrary  $w \in \Gamma^*$ . Describe a derivation of  $w_T$  by  $G$  using  $v_0, v_1, \dots, v_d \in \Sigma_\Gamma$ ,  $d \geq 1$ , where

$$\begin{aligned}
 v_0 &= A_S, \\
 v_d &= w_T, \\
 v_k &= v_{\mathbf{p},k} A_{i_k} v_{\mathbf{s},k}, \\
 v_{k+1} &= v_{\mathbf{p},k} u_{i_k} v_{\mathbf{s},k}
 \end{aligned}$$

for each  $k \in \{0, \dots, d-1\}$ . For  $k \in \{0, \dots, d\}$ , we define inductively a word  $w_k \in \Sigma_\Gamma$  satisfying  $(w_k)_T = v_k$  as follows. First,  $w_0 = A_S$ . Next, take  $0 \leq k \leq d-1$  and write  $w_k = w_{\mathbf{p},k} A_{i_k} w_{\mathbf{s},k}$  such that  $(w_{\mathbf{p},k})_T = v_{\mathbf{p},k}$  and  $(w_{\mathbf{s},k})_T = v_{\mathbf{s},k}$ . Then define

$$w_{k+1} = w_{\mathbf{p},k} A_{i_k} b_{i_k} u_{i_k} w_{\mathbf{s},k}.$$

Informally, the words  $w_0, \dots, w_d$  describe the derivation of  $w_T$  with keeping all the used nonterminals, i.e.,  $A_{i_k}$  is rewritten by  $A_{i_k} b_{i_k} u_{i_k}$  instead of  $u_{i_k}$ . Observe

<sup>1</sup> A *projection* to  $\Sigma \subseteq \Gamma$  is given by the homomorphism that maps  $x \in \Gamma$  to  $x$  if  $x \in \Sigma$  or to  $\epsilon$  otherwise.

that  $q_1 w_d \curvearrowright^* q_1 A_S$  using the transitions labeled by words from  $P_{BU}$ . Also observe that, due to Greibach normal form,  $w_d \in (\Sigma_T \cup \Sigma_T \Sigma_N \Sigma_B)^*$ , which means that the factors from  $\Sigma_N \Sigma_B$  are always separated with letters from  $\Sigma_T$ .

Distinguish the following cases:

- If  $w$  does not have a factor from  $\Gamma^2 \setminus P_C$ , all two-letter factors of  $w$  belong to  $P_C$ , which implies that  $w$  is a factor of some word from  $(\Sigma_T t)^*$ , where

$$t = A_1 b_1 A_2 b_2 \cdots A_m b_m. \quad (2)$$

- If  $w$  begins with a letter from  $\Sigma_T \cup \Sigma_N$ , and ends with a letter from  $\Sigma_T \cup \Sigma_B$ , then  $q_1 w \curvearrowright^* q_1 w_d$  using the transitions labeled by words from  $P_{NB}$ . Because  $q_1 w_d \curvearrowright^* q_1 A_S$ , we conclude  $w \in L(M_G)$ .
- Otherwise,  $w$  starts with a letter from  $\Sigma_B$  or ends with a letter from  $\Sigma_N$ . Then

$$w_{N,B} \in \Sigma_B (\Sigma_N \Sigma_B)^* \cup (\Sigma_N \Sigma_B)^* \Sigma_N \cup \Sigma_B (\Sigma_N \Sigma_B)^* \Sigma_N$$

and we observe that  $q_0 w \curvearrowright q_3 w \curvearrowright^* q_3 w_{N,b} \curvearrowright u$  for some  $u \in \Sigma_N \cup \Sigma_b \cup \{\epsilon\}$ . As  $q_3 u \curvearrowright q_4$ , we get  $w \in L(M_G)$ .

- If  $w$  has a factor  $u \subseteq \Gamma^2 \setminus P_C$ , write  $w = w_p u w_s$  and observe

$$w_p q_0 u w_s \curvearrowright q_2 w_p w_s \curvearrowright^* q_2,$$

implying  $w \in L(M_G)$ .

Second, suppose that  $L(M_G) = \Gamma^*$  and take an arbitrary  $v = x_1 x_2 \cdots x_n \in \Sigma_T^*$  with  $x_1, \dots, x_n \in \Sigma_T$ . Let  $w = (x_1 t)(x_2 t) \cdots (x_{n-1} t)(x_n t)$ , with  $t$  defined in (2). We have  $w \in L(M_G)$ . Observe that:

- The word  $w$  does not contain a factor from  $\Gamma^2 \setminus P_C$ .
- By deleting factors from  $\Sigma_B \Sigma_N \cup \Sigma_T$ , the word  $w$  cannot become a word from  $\Sigma_N \cup \Sigma_B \cup \{\epsilon\}$ .

Thus,  $w$  can be accepted by  $M$  only using a path through the state  $q_1$  ending in the state  $q_4$ . In other words,  $w$  can be obtained by inserting words from  $P_{BU} \cup P_{NB}$  to  $A_S$ . During that process, once an occurrence of some  $b_i$  fails to be preceded by  $A_i$ , this situation lasts to the very end, which is a contradiction. It follows that  $b_i u_i \in P_{BU}$  can be inserted only to the right of an occurrence of  $A_i$  that was not followed by  $b_i$  yet. This corresponds to rewriting  $A_i$  with  $u_i$  and we can observe that  $w_T = v$  is necessarily generated by the grammar  $G$ .

## 4 Clearing Restarting Automata with Small Contexts

Though the basic model of clearing restarting automata is not able to describe all context-free languages nor to handle basic language operations (e.g. concatenation and union) [12], it has been deeply studied in order to design suitable generalizations. The study considered also restrictions of the maximum context length to be used in rewriting rules:

**Theorem 6 ([12]).**

1. For each  $k \geq 3$ , the class  $\mathcal{L}(k\text{-cl-RA})$  contains a binary language, which is not context-free.
2. The class  $\mathcal{L}(2\text{-cl-RA})$  contains a language  $L \subseteq \Sigma^*$  with  $|\Sigma| = 6$ , which is not context-free.
3. The class  $\mathcal{L}(k\text{-cl-RA})$  contains only context-free languages.

The present section is devoted to proving the following theorem, which completes the results listed above.

**Theorem 7.** *The class  $\mathcal{L}(2\text{-cl-RA})$  contains a binary language, which is not context-free.*

In order to prove Theorem 7, we define two particular rewriting systems:

1. A 1-context rewriting system  $R_{uV} = (\{u, V\}, \{u, V\}, I_{uV})$ . The set  $I_{uV}$  is listed in Table 1.
2. A 2-clearing restarting automaton  $R_{01} = (\{0, 1\}, I_{01})$ . The set  $I_{01}$  is listed in Table 2.

We write  $\rightarrow_{uV}$  for the rewriting relation of  $R_{uV}$  and  $\vdash_{01}$  for the production relation of  $R_{01}$ .

0)	$(\dot{c}, \epsilon \rightarrow uu, \$)$
1)	$(\dot{c}, u \rightarrow uuV, \epsilon)$
2)	$(\epsilon, Vu \rightarrow uuuV, \epsilon)$
3)	$(\epsilon, Vu \rightarrow uuuu, \$)$

**Table 1.** The rules  $I_{uV}$

	(a)	(b)	(c)	(d)
0)	$(\dot{c}, 00, \$)$	-	-	-
1)	$(\dot{c}, 10, 00)$	$(\dot{c}, 00, 10)$	-	-
2)	$(01, 10, 00)$	$(00, 11, 01)$	$(11, 00, 10)$	$(10, 01, 11)$
3)	$(01, 10, 0\$)$	$(00, 11, 0\$)$	-	-

**Table 2.** The rules  $I_{01}$  sorted by types 0 to 3

The key feature of the system  $R_{uV}$  is:

**Lemma 8.** *Let  $w \in L(R_{uV}) \cap \{u\}^*$ . Then  $|w| = 2 \cdot 3^n$  for some  $n \geq 0$ .*

The proof is postponed to Section 4.1. We also define:

1. A length-preserving mapping  $\varphi : \{0, 1\}^* \rightarrow \{u, V\}^*$  as  $\varphi(x_1 \dots x_n) = \bar{x}_1 \dots \bar{x}_n$ , where

$$\bar{x}_k = \begin{cases} V & \text{if } 1 < k < n \text{ and } x_{k-1} = x_{k+1} \\ u & \text{otherwise} \end{cases}$$

for each  $k \in \{1, \dots, n\}$ .

2. A regular language  $K \subseteq \{0, 1\}^*$ :

$$K = \{w \in \{0, 1\}^* \mid w \text{ has none of the factors } 000, 010, 101, 111\}.$$

The following is a trivial property of  $\varphi$  and  $K$ :

**Lemma 9.** *Let  $u \in \{0, 1\}^*$ . Then  $u \in K$  if and only if  $\varphi(u) \in \{u\}^*$ .*

The next lemma expresses how the systems  $R_{01}$  and  $R_{uV}$  are related:

**Lemma 10.** *Let  $u, v \in \{0, 1\}^*$ . If  $u \dashv_{01} v$ , then  $\varphi(u) \rightarrow_{uV} \varphi(v)$ .*

*Proof.* For  $u = v$  the claim is trivial, so we suppose  $u \neq v$ . Denote  $m = |u|$ . As  $u$  can be rewrote to  $v$  using a single rule of  $R_{01}$ , we can distinguish which of the four kinds of rules (the rows 0 to 3 of Table 2) is used:

- 0) If the rule 0 is used, we have  $u = \epsilon$  and  $v = 00$ . Thus  $\varphi(u) = \epsilon$  and  $\varphi(v) = uu$ .
- 1) If a rule  $(\dot{c}, z_1 z_2, y_1 y_2)$  of the kind 1 is used, we see that  $v$  has some of the prefixes 1000, 0010 and so  $\varphi(v)$  starts with  $uuV$ . Trivially,  $\varphi(u)$  starts with  $u$ . Because  $u[1..] = v[3..]$ , we have  $\varphi(u)[2..] = \varphi(v)[4..]$  and we conclude that applying the rule  $(\dot{c}, u \rightarrow uuV, \epsilon)$  rewrites  $\varphi(u)$  to  $\varphi(v)$ .
- 2) If a rule  $(x_1 x_2, z_1 z_2, y_1 y_2)$  of the kind 2 is used, we have

$$\begin{aligned} u[k..k+3] &= x_1 x_2 y_1 y_2, \\ v[k..k+5] &= x_1 x_2 z_1 z_2 y_1 y_2. \end{aligned}$$

for some  $k \in \{1, \dots, m-3\}$ . As  $x_1 x_2 y_1 y_2$  equals some of the factors 0100, 0001, 1110, 1011, we have

$$\varphi(u)[k+1..k+2] = Vu.$$

As  $x_1 x_2 z_1 z_2 y_1 y_2$  equals some of the factors 011000, 001101, 110010, 100111, we have

$$\varphi(v)[k+1..k+4] = uuuV.$$

Because  $u[.. $k+1]$  =  $v[.. $k+1]$  and  $u[k+2..] = v[k+4..]$ , we have$$

$$\begin{aligned} \varphi(u)[..k] &= \varphi(v)[..k], \\ \varphi(u)[k+3..] &= \varphi(v)[k+5..]. \end{aligned}$$

Now it is clear that the rule  $(\epsilon, Vu \rightarrow uuuV, \epsilon)$  rewrites  $\varphi(u)$  to  $\varphi(v)$ .

- 3) If a rule  $(x_1 x_2, z_1 z_2, y\$)$  of the kind 3 is used, we have

$$\begin{aligned} u[m-2..m] &= x_1 x_2 y, \\ v[m-2..m+2] &= x_1 x_2 z_1 z_2 y. \end{aligned}$$

As  $x_1 x_2 y$  equals some of the factors 010, 000, we have

$$\varphi(u)[m-1..m] = Vu.$$

As  $x_1 x_2 z_1 z_2 y$  equals some of the factors 01100, 00110, we have

$$\varphi(v)[m-1..m+2] = uuuV.$$

Because  $u[..m-1] = v[..m-1]$ , we have

$$\varphi(u)[..m-2] = \varphi(v)[..m-2],$$

Now it is clear that the rule  $(\epsilon, \forall u \rightarrow uuuu, \$)$  rewrites  $\varphi(u)$  to  $\varphi(v)$ .

**Corollary 11.** *If  $u \in L(R_{01})$ , then  $\varphi(u) \in L(R_{uv})$ .*

*Proof.* Follows from the fact that  $\varphi(\epsilon) = \epsilon$  and a trivial inductive use of Lemma 10.

The last part of the proof of Theorem relies on the following lemma, whose proof is postponed to Section 4.1:

**Lemma 12.** *For each  $\alpha, \beta > 0$  it holds that*

$$00(1100)^\alpha 1000(1100)^\beta \dashv_{01} 00(1100)^{\alpha+9} 1000(1100)^{\beta-1}.$$

**Corollary 13.** *For each  $\beta > 0$  it holds that*

$$001000(1100)^\beta \dashv_{01} 00(1100)^{9\beta} 1000.$$

*Proof.* As the left-hand side is equal to  $00(1100)^0 1000(1100)^\beta$  and the right-hand side is equal to  $00(1100)^{9\beta} 1000(1100)^0$ , the claim follows from an easy inductive use of Lemma 12.

**Corollary 14.** *The language  $L(R_{01}) \cap K$  is infinite.*

*Proof.* We show that for each  $k \geq 0$ ,

$$00(1100)^{\frac{2 \cdot 9^k - 2}{4}} \in L(R_{01}).$$

In the case  $k = 0$  we just check that  $00 \in L(R_{01})$ . Next we suppose that the claim holds for a fixed  $k \geq 0$  and show that

$$00(1100)^{\frac{2 \cdot 9^k - 2}{4}} \dashv_{01} 00(1100)^{\frac{2 \cdot 9^{k+1} - 2}{4}}.$$

Using the rules 1a and 1b we get

$$00(1100)^{\frac{2 \cdot 9^k - 2}{4}} \dashv_{01} 1000(1100)^{\frac{2 \cdot 9^k - 2}{4}} \dashv_{01} 001000(1100)^{\frac{2 \cdot 9^k - 2}{4}},$$

while Corollary 13 continues with

$$001000(1100)^{\frac{2 \cdot 9^k - 2}{4}} \dashv_{01} 00(1100)^{\frac{2 \cdot 9^{k+1} - 18}{4}} 1000.$$

Finally, denoting  $p = 00(1100)^{\frac{2 \cdot 9^{k+1} - 18}{4}}$ , using rules 2b, 2a, 2b, 2d, 2c, and 2a respectively we get

$$\begin{aligned} & p1000 \dashv_{01} p100\underline{11}0 \dashv_{01} p1\underline{1000}110 \dashv_{01} p1100\underline{11}0110 \dashv_{01} p11001100\underline{11}10 \dashv_{01} \\ & \dashv_{01} p1100110011\underline{00}10 \dashv_{01} p1100110011001\underline{100} = 00(1100)^{\frac{2 \cdot 9^{k+1} - 2}{4}}. \end{aligned}$$



We conclude the proof of Theorem 7 by pointing out that Lemmas 9, 10, and 8 say that for each  $w \in \{0, 1\}^*$  we have

$$\begin{aligned} w \in L(R_{01}) \cap K &\Rightarrow \varphi(w) \in L(R_{uV}) \cap \{u\}^* \\ &\Rightarrow (\exists n \geq 0) |w| = 2 \cdot 3^n \end{aligned}$$

This, together with the pumping lemma for context-free languages and the infiniteness of  $L(R_{01}) \cap K$ , implies that  $L(R_{01}) \cap K$  is not a context-free language. As the class of context-free languages is closed under intersections with regular languages, nor  $L(R_{01})$  is context-free.

#### 4.1 Proofs of Lemmas 8 and 12

*Proof (of Lemma 8).* We should prove that  $w \in L(R_{uV}) \cap \{u\}^*$  implies  $|w| = 2 \cdot 3^n$  for some  $n \geq 0$ . Let  $\Phi : \{u, V\}^* \rightarrow \mathbb{N}$  be defined inductively as follows:

$$\begin{aligned} \Phi(\epsilon) &= 0, \\ \Phi(u^k w) &= k + \Phi(w), \\ \Phi(Vw) &= 1 + 3 \cdot \Phi(w) \end{aligned}$$

for each  $k \geq 1$  and  $w \in \{u, V\}^*$ . Observe that we have assigned a unique value of  $\Phi$  to each word from  $\{u, V\}^*$ . Next, we describe effects of the rules of  $R_{uV}$  to the value of  $\Phi$ .

- 0) The rule 0 can only rewrite  $w_1 = \epsilon$  to  $w_2 = uu$ . We have  $\Phi(w_1) = 0$  and  $\Phi(w_2) = 2$ .
- 1) The rule 1 rewrites  $w_1 = uw$  to  $w_2 = uuVw$  for some  $w \in \{u, V\}^*$ . We have  $\Phi(w_1) = 1 + \Phi(w)$  and  $\Phi(w_2) = 3 + 3 \cdot \Phi(w)$ . Thus,  $\Phi(w_2) = 3 \cdot \Phi(w_1)$ .
- 2) The rule 2 rewrites  $w_1 = \overline{w}Vu$  to  $w_2 = \overline{w}uuuVw$  for some  $w, \overline{w} \in \{u, V\}^*$ . We have

$$\Phi(Vuw) = \Phi(uuuVw) = 4 + 3 \cdot \Phi(w).$$

It follows that  $\Phi(w_1) = \Phi(w_2)$ .

- 3) The rule 3 rewrites  $w_1 = \overline{w}Vu$  to  $w_2 = \overline{w}uuuu$  for some  $\overline{w} \in \{u, V\}^*$ . We have  $\Phi(Vu) = \Phi(uuuu) = 4$  and thus  $\Phi(w_1) = \Phi(w_2)$ .

Together, each  $w \in L(R_{uV})$  has  $\Phi(w) = 2 \cdot 3^n$  for some  $n \geq 0$ . As  $\Phi(w) = |w|$  for each  $w \in \{u\}^*$ , the proof is complete.

*Proof (of Lemma 12).* We should prove that

$$00(1100)^\alpha 1000(1100)^\beta \dashv_{01} 00(1100)^{\alpha+9} 1000(1100)^{\beta-1}.$$

for each  $\alpha, \beta > 0$ . Indeed:

$$\begin{array}{llll}
00(1100)^\alpha 1000(1100)^\beta & \neg_{01} & 00(1100)^\alpha 100\underline{110}(1100)^\beta & \neg_{01} \\
00(1100)^\alpha \underline{11000}110(1100)^\beta & \neg_{01} & 00(1100)^{\alpha+1} \underline{110}110(1100)^\beta & \neg_{01} \\
00(1100)^{\alpha+1} 1100\underline{1110}(1100)^\beta & \neg_{01} & 00(1100)^{\alpha+2} 1110\underline{01}(1100)^\beta & \neg_{01} \\
00(1100)^{\alpha+2} \underline{1100}1001(1100)^\beta & \neg_{01} & 00(1100)^{\alpha+3} \underline{11000}1(1100)^\beta & \neg_{01} \\
00(1100)^{\alpha+4} \underline{1101}(1100)^\beta & \neg_{01} & 00(1100)^{\alpha+4} 1101\underline{100}100(1100)^{\beta-1} & \neg_{01} \\
00(1100)^{\alpha+4} \underline{11001}1100100(1100)^{\beta-1} & \neg_{01} & 00(1100)^{\alpha+5} \underline{1100}100100(1100)^{\beta-1} & \neg_{01} \\
00(1100)^{\alpha+6} \underline{11000}100(1100)^{\beta-1} & \neg_{01} & 00(1100)^{\alpha+7} 01\underline{1000}(1100)^{\beta-1} & \neg_{01} \\
00(1100)^{\alpha+7} \underline{1101}1000(1100)^{\beta-1} & \neg_{01} & 00(1100)^{\alpha+7} 1100\underline{11}1000(1100)^{\beta-1} & \neg_{01} \\
00(1100)^{\alpha+8} \underline{1100}1000(1100)^{\beta-1}. & & & 
\end{array}$$

## References

1. Ehrenfeucht, A., Haussler, D., Rozenberg, G.: On regularity of context-free languages. *Theoretical Computer Science* 27(3), 311 – 332 (1983)
2. Fernau, H., Paramasivan, M., Schmid, M.L.: Characterization and complexity results on jumping finite automata. submitted to *Theoretical Computer Science* (2015)
3. Fernau, H., Paramasivan, M., Schmid, M.L.: Jumping finite automata: Characterizations and complexity. In: Drewes, F. (ed.) *Implementation and Application of Automata*, *Lecture Notes in Computer Science*, vol. 9223, pp. 89–101. Springer International Publishing (2015)
4. Haussler, D.: Insertion languages. *Information Sciences* 31(1), 77 – 89 (1983)
5. Hopcroft, J.E., Motwani, R., Ullman, J.D.: *Introduction to automata theory, languages, and computation*, 2nd edition. Addison-Wesley (2003)
6. Ito, M., Kari, L., Thierrin, G.: Insertion and deletion closure of languages. *Theoretical Computer Science* 183(1), 3 – 19 (1997)
7. Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata. In: Reichel, H. (ed.) *Fundamentals of Computation Theory*, *Lecture Notes in Computer Science*, vol. 965, pp. 283–292. Springer Berlin Heidelberg (1995)
8. Meduna, A., Zemek, P.: Jumping finite automata. *International Journal of Foundations of Computer Science* 23(7), 1555–1578 (2012)
9. Meduna, A., Zemek, P.: *Regulated Grammars and Automata*. Springer US (2014), chapter 17: Jumping Finite Automata
10. Mráz, F., Plátek, M., Vogel, J.: Restarting automata with rewriting. In: Jeffery, K., Král, J., Bartošek, M. (eds.) *SOFSEM’96: Theory and Practice of Informatics*, *Lecture Notes in Computer Science*, vol. 1175, pp. 401–408. Springer Berlin Heidelberg (1996)
11. Černo, P.: Clearing restarting automata and grammatical inference. In: Jeffrey Heinz, Colin de la Higuera, T.O. (ed.) *Proceedings of the Eleventh International Conference on Grammatical Inference*. *JMLR Workshop and Conference Proceedings*, vol. 21, pp. 54–68 (2012)
12. Černo, P., Mráz, F.: Clearing restarting automata. *Fundamenta Informaticae* 104(1), 17–54 (2010)

13. Vorel, V.: On basic properties of jumping finite automata. International Journal of Foundations of Computer Science (conditionally accepted) (2015)