Pumping Deterministic Monotone Restarting Automata and DCFL

František Mráz[1], Dana Pardubská[2]*, Martin Plátek[1]†, and Jiří Šíma[3]‡

[1] Charles University, Department of Computer Science
Malostranské nám. 25, 118 00 PRAHA 1, Czech Republic
martin.platek@mff.cuni.cz, frantisek.mraz@mff.cuni.cz

[2] Comenius University in Bratislava, Department of Computer Science
Mlynská Dolina, 84248 Bratislava, Slovakia
pardubska@dcs.fmph.uniba.sk

[3] Institute of Computer Science of the Czech Academy of Sciences,
P. O. Box 5, 18207 Prague 8, Czech Republic
sima@cs.cas.cz

Abstract: We introduce a new type of the deterministic monotone restarting automaton that enables new types of characterization of the class of deterministic context-free languages (DCFL) based on pumping. The characterization is obtained through new types of normalizations of deterministic monotone restarting automata. This paper is the first step to prepare notions for studying the relation between restarting automata and analog neuron automata, and for studying degrees of non-regularity of DCFL.

1 Introduction

Restarting automata were introduced in [3] as a linguistically motivated model of automata that enables to study so-called analysis by reduction of a natural language. An overview of several variants of the model can be found in [6].

The original model of restarting automata denoted as R-automaton is a finite state machine equipped with a read/write window of a fixed length k that can move over a flexible tape. The word on its tape is always delimited by a pair of sentinels ¢ and \$. The automaton works in cycles. Each cycle starts in the initial state with the window on the left end of the tape so it scans the left sentinel ¢ and the first $k-1$ symbols of the current tape contents, or the rest of the tape (when the word is shorter than $k-2$). Then the automaton moves from left to right while changing states according to its transition function until it decides to rewrite the tape by deleting some symbols scanned by its window together with removing the cells of the tape containing the deleted symbols. Immediately after a rewrite the automaton "restarts" the computation on its shortened tape, that is, it enters its initial state and places the read/write window at the leftmost position again. A computation of an R-automaton ends when during scanning the tape it enters a halting state that can be either accepting, in which case it accepts the input word, or rejecting, in which case it rejects the input word.

The first fundamental result on restarting automata was a characterization of the class of deterministic context-free languages (DCFL) by a subclass of deterministic monotone R-automata. A computation of an R-automaton is monotone if the distances between the places of rewriting and the right sentinel are decreasing (maybe not strictly) during the whole computation. An R-automaton is monotone if all its computations are monotone.

An essential part of this paper is derived from [3] and [5]. We study the so-called RP-automata that slightly differ from R-automata in [3] and RW-automata in [4]. One restarting step by RW-automata is by RP-automata substituted by two consecutive steps: by a preparing step, and by a restarting step. With such a modification it is easier to present their so-called pumping properties.

The paper is structured as it follows. The next section introduces the model of RP-automata and its deterministic and monotone variant, and states some basic properties of the model. Section 3 introduces pumping instructions and pumping restarting automata. Pumping rewriting instructions correspond roughly to "pumping" used in the pumping lemma for context-free languages (cf. [2]) and pumping restarting automata are RP-automata that have only pumping rewriting instructions. We show there that using a strong cyclic form of deterministic RP-automata, we can check whether a given rewriting instruction is pumping by inspecting only computations on words of length limited by a constant. It follows new characterizations of DCFL by deterministic pumping restarting automata. Finally, we show conditions ensuring that a pumping instruction causes that an RP-automaton accepts a non-regular language.

2 Definitions and Results

A *restarting automaton of type P*, or an RP-*automaton*, $M = (Q, \Sigma, ¢, \$, q_0, k, \delta, Q_A, Q_R)$ (with k-bounded lookahead) is a device with a finite state control unit with the

*The research is partially supported by VEGA 1/0601/20

†The research was partially supported by the grant of the Czech Science Foundation GA19-05704S during the author's stay at Institute of Computer Science, Czech Academy of Sciences.

‡The research is partially supported by the grant of the Czech Science Foundation GA19-05704S

finite set of states Q containing two disjunctive subsets Q_{A}, Q_{R} of accepting and rejecting states, respectively. The automaton is equipped with a head moving on a finite linear flexible tape of items (cells). The first item of the tape contains always the left sentinel symbol ¢, the last one the right sentinel symbol \$, and each other item contains a symbol from a finite alphabet Σ (not containing ¢, \$). The head has a flexible read/write window of length at most k (for some $k \geq 1$) – M scans k consecutive items or the rest of the tape when the distance to the right sentinel \$ is less than k. We say that M is of window size k. In the *initial configuration* on an input word $w \in \Sigma^*$, the tape contains the input word delimited by the sentinels ¢ and \$, the control unit is in the initial state q_0, and the window scans the left sentinel ¢ and the first $k-1$ symbols of the input word (or the rest of the tape if the tape contents is shorter than k).

The *computation* of M is controlled by the *transition function*

$$\delta : (Q \setminus (Q_{\mathrm{A}} \cup Q_{\mathrm{R}})) \times \mathscr{PC}^{\leq(k)} \rightarrow \mathscr{P}(Q \times \{MVR, PREPARE\}) \cup \{RESTART(v) \mid v \in \mathscr{PC}^{\leq(k-1)}\}.$$

Here $\mathscr{P}(S)$ denotes the powerset of the set S, $\mathscr{PC}^{(k)}$ is the set of *possible contents* of the read/write window of M, where for $i, n \geq 0$

$$\mathscr{PC}^{(i)} := (\{¢\} \cdot \Sigma^{i-1}) \cup \Sigma^i \cup (\Sigma^{\leq i-1} \cdot \{\$\}) \cup (\{¢\} \cdot \Sigma^{\leq i-2} \cdot \{\$\}),$$

$$\Sigma^{\leq n} := \bigcup_{i=0}^{n} \Sigma^i \quad \text{and} \quad \mathscr{PC}^{\leq(k)} := \bigcup_{i=0}^{k} \mathscr{PC}^{(i)}.$$

The transition function represents a finite set of four different types of instructions (transition steps). Let q, q', q_I be states from Q, $u \in \mathscr{PC}^{(k)}$, $w \in \mathscr{PC}^{\leq(k)}$, $v \in \mathscr{PC}^{\leq(k-1)}$ and M be in state q with u be the contents of its read/write window:

(1) A *move-right instruction* of the form $(q,u) \rightarrow_\delta (q', MVR)$ is applicable if $u \neq \$$. It causes M to enter the state q' and to move its read/write head one item to the right.

(2) *A preparing instruction* $(q,u) \rightarrow_\delta (q_I, PREPARE)$ changes M's state to a restarting state q_I that determines the next instruction, which must be a restarting instruction.

(3) *A restarting instruction I* is of the form $(q_I, w) \rightarrow_\delta RESTART(v)$, where $|v| < |w|$ and if w contains any sentinel, v contains the corresponding sentinels, too. This instruction is applicable if w is a prefix of the contents of the read/write window. When executed, M replaces w with v (hereby it shortens its tape) and restarts – i.e. it enters the initial state and places the window at the leftmost position so that the first item in the window contains ¢. Note that the pair (w,v) is unambiguously given by the state q_I. We can assume that all pairs (w,v) such that $|v| < |w|$ and the word w can be replaced with v by some *RESTART* instruction are ordered and that I is the index of (w,v) in that sequence. Thus, although RP-automaton is in general nondeterministic, each *RESTART* instruction of M corresponds unambiguously to one restart state q_I.

(4) *A halting instruction* of the form $(q,u) \rightarrow_\delta (q', HALT)$, where $q' \in Q_{\mathrm{A}}$ or $q' \in Q_{\mathrm{R}}$, finishes the computation and causes M to accept or reject, respectively, the input word.

Thus, the set of states can be divided into three groups – the *halting states* $Q_{\mathrm{A}} \cup Q_{\mathrm{R}}$, the *restarting states* (involved at left-hand side of restarting instructions) and the rest, called the *transition states*.

A *configuration* of an RP-automaton M is a word $\alpha q \beta$, where $q \in Q$, and either $\alpha = \lambda$ and $\beta \in \{¢\} \cdot \Sigma^* \cdot \{\$\}$ or $\alpha \in \{¢\} \cdot \Sigma^*$ and $\beta \in \Sigma^* \cdot \{\$\}$; here q represents the current state, $\alpha\beta$ is the current contents of the tape, and it is understood that the read/write window contains the first k symbols of β or all symbols of β if $|\beta| < k$. An *initial (restarting) configuration* is of the form $q_0 ¢ w \$$, where $w \in \Sigma^*$. A *rewriting configuration* is of the form $\alpha q_I \beta$, where q_I is a restarting state.

A *computation* of M is a sequence $C = C_0, C_1, \ldots, C_j$ of configurations of M, where C_0 is a restarting configuration and $C_{\ell+1}$ is obtained from C_ℓ by a step of M, for all ℓ, $0 \leq \ell < j$, denoted as $C_\ell \vdash_M C_{\ell+1}$ and $\vdash_M^*$ is the reflexive and transitive closure of the single step relation $\vdash_M$.

In general, an RP-automaton can be *nondeterministic*, i.e. there can be two or more instructions with the same left-hand side. If that is not the case, the automaton is *deterministic*. In what follows we are mostly interested in deterministic RP-automata, denoted det-RP.

An input *word w is accepted by M* if there is a computation that starts in the initial configuration with w (bounded by sentinels ¢, \$) on the tape and finishes in an *accepting configuration* where the control unit is in one of the accepting states. $L(M)$ denotes the language consisting of all words accepted by M; we say that M *accepts the language* $L(M)$.

Restarting steps divide any computation of an RP-automaton into certain phases that all start in the initial state in restarting configurations with the read/write window in the leftmost position. In a phase called *cycle*, the head moves to the right along the input list (with its read-/write window) until a restart occurs – in that case the computation is resumed in the initial configuration on a new, shorter, word. The phase from the last restart to the halting configuration is called *tail*. This immediately implies that any computation of any RP-automaton is finite (ending in a halting state).

The next proposition expresses certain lucidness of computations of deterministic RP-automata. The notation $u \Rightarrow_M v$ means that there exists a cycle of M starting in the initial configuration with the word u on its tape and finishing in the initial configuration with the word v on its tape; the relation $\Rightarrow_M^*$ is the reflexive and transitive closure of $\Rightarrow_M$.

The validity of the following proposition is obvious.

Proposition 1. (Correctness preserving property.) *Let M be a deterministic* RP*-automaton and $u \Rightarrow_M^* v$ for some*

words u, v. Then $u \in L(M)$ iff $v \in L(M)$.

By a *monotone* RP-*automaton* we mean an RP-automaton where the following holds for all computations: all items which appeared in the read/write window (and remained still on the tape) during one cycle will appear in the read/write window in the next cycle as well. It means, that during any computation of monotone RP-automaton the items from the read/write window of prepare configurations do not increase their distances from the right endmarker \$.

Considering a deterministic RP-automaton M, it is for us convenient to suppose it to be in the *strong cyclic form*; it means that the words of length less than k, k being the length of its read/write window, are immediately (hence in the tail) accepted or rejected, and that M performs at least one cycle (at least one restarting) on any longer word.

We use the following obvious notation. RP denotes the class of all (nondeterministic) RP-automata. Prefix det- denotes the deterministic version, similarly mon- the monotone version. Prefix scf- denotes the version in the strong cyclic form. $\mathscr{L}(A)$, where A is some class of automata, denotes the class of languages accepted by automata from A. E.g., the class of languages accepted by deterministic monotone RP-automata is denoted by $\mathscr{L}$(det-mon-RP).

Since all computations of RP-automata are finite, the following proposition is obvious.

Proposition 2. *The classes $\mathscr{L}$(det-mon-RP) and $\mathscr{L}$(det-RP) are closed under complement.*

Rejected languages by det-RP**-automata.** Let M be a det-RP-automaton, and $L(M) = L$. We say that the language $\overline{L}$ (the complement of L) is rejected by M. We will often use the fact that the language and its complement can be recognized (distinguished) by the same det-RP-automaton.

The natural question of the decidability of monotonicity for a given RP-automaton is answered in the affirmative:

Theorem 1. *There is an algorithm which, given an* RP-*automaton M, decides whether M is monotone or not.*

Proof: The proof is the same as the corresponding proof from [4]. The only difference between RW- and RP-automaton is that the *RESTART* operation of RW-automaton is unambiguously split into two consecutive instructions *PREPARE* and *RESTART*. This splitting has no influence on the decidability proof, since whenever the *RESTART* operation of RW-automaton is used/simulated in the original proof it can unambiguously be replaced by *PREPARE* and *RESTART* operation of RP-automaton. □

In what follows, we consider only deterministic RP-automata. We write $\alpha \Rightarrow_I \beta$, for words $\alpha, \beta \in \Sigma^*$ when α was shortened to β in one cycle consisting of several *MVR* steps followed by a *PREPARE* instruction and the restarting instruction $I = (q_I, w) \to_\delta RESTART(v)$ with the restart state q_I. Additionally, when $\text{¢}xq_Iwy\$$ is the rewriting configuration corresponding to the reduction $xwy \Rightarrow_I xvy$, for some words x, y, v, w, we will underline the occurrence of w rewritten in the cycle. That is, we will write $x\underline{w}y \Rightarrow_I xvy$. Analogously, $\alpha \Rightarrow_I^* \beta$ indicates a computation consisting of several such cycles that use the same restarting instruction I with the restarting state q_I.

3 Pumping Restarting Automata

Let $M = (Q, \Sigma, \text{¢}, \$, q_0, k, \delta, Q_A, Q_R)$ be a det-mon-RP-automaton, and $\iota = (q_\iota, w) \to_\delta RESTART(v)$ a restarting instruction of M. We say that ι is a *pumping instruction* of M if w can be written as $w = u_1vu_2$, for some words u_1, u_2, $u_1 \neq \lambda$, and, for all $x, y \in \Sigma^*$, $xu_1\underline{vu_2}y \Rightarrow_\iota xvy$ implies

$$xu_1^{j-1}\underline{u_1vu_2}u_2^{j-1}y \Rightarrow_\iota xu_1^{j-1}vu_2^{j-1}y, \text{ for all } j \geq 1.$$

We say that ι has pumping words u_1, u_2. If $u_2 \neq \lambda$ we say that ι is a *two-side pumping instruction*. If $u_2 = \lambda$ we say that ι is a *one-side pumping instruction*.

Note that a preparing instruction determines whether a given restarting instruction can be executed and each preparing instruction depends on move right instructions that are executed before it. Hence, the property of being a pumping instruction depends on the whole transition function of an RP-automaton.

Let M be a det-mon-RP-automaton and all restarting instructions of M are pumping instructions. Then we call M a *pumping* RP-*automaton*. We denote the property of pumping by the prefix pmp-.

Example 1. *Let $M_1 = (Q, \Sigma, \text{¢}, \$, q_0, k, \delta, Q_A, Q_R)$ be the* RP-*automaton with the set of states $Q = \{q_0, q_1, q_A, q_R\}$, the alphabet $\Sigma = \{a, b\}$, window size $k = 2$, the set of accepting states $Q_A = \{q_A\}$, the set of rejecting states $Q_R = \{q_R\}$ and the transition function*
$\delta(q_0, \text{¢}\$) = \{(q_A, HALT)\}$, $\delta(q_0, bb) = \{(q_0, MVR)\}$,
$\delta(q_0, \text{¢}a) = \{(q_0, MVR)\}$, $\delta(q_0, ab) = \{(q_1, PREPARE)\}$,
$\delta(q_0, \text{¢}b) = \{(q_0, MVR)\}$, $\delta(q_1, ab) = \{RESTART(\lambda)\}$,
$\delta(q_0, aa) = \{(q_0, MVR)\}$, $\delta(q_0, a\$) = \{(q_R, HALT)\}$,
$\delta(q_0, ba) = \{(q_0, MVR)\}$, $\delta(q_0, b\$) = \{(q_R, HALT)\}$.
The automaton has only one restarting instruction $\iota = \delta(q_1, ab) \to_\delta RESTART(\lambda)$. This instruction is two-side pumping instruction as $w = ab$ can be written as u_1vu_2, where $u_1 = a$ and $u_2 = b$ are nonempty, $v = \lambda$. If $q_0\text{¢}xu_1vu_2y\$ = q_0\text{¢}xaby\$ \vdash^* \text{¢}xq_1u_1vu_2y\$ \vdash_\iota q_0\text{¢}xvy\$ = q_0\text{¢}xy\$$ for some $x, y \in \{a, b\}^*$, then it holds that $xu_1^jvu_2^jy = xa^jb^jy \Rightarrow_\iota^* xy = xvy$, for all $j \geq 0$, and $xu_1^{j-1}\underline{u_1vu_2}u_2^{j-1}y \Rightarrow_\iota xu_1^{j-1}vu_2^{j-1}y$, for all $j > 0$. Evidently, the automaton M accepts the Dyck language of correctly paired parentheses, where a and b are the left and right parenthesis, respectively.

Note that M_1 is not in the strong cycling form.

Example 2. *Consider the regular language L_2 of even length words over the alphabet $\{a\}$. This language can*

easily be accepted by the following RP*-automaton* M_2*,* $M_2 = (Q,\Sigma,¢,\$,q_0,k,\ \delta,Q_A,Q_R)$ *with the set of states* $Q = \{q_0,q_1,q_A,q_R\}$*, the alphabet* $\Sigma = \{a\}$*, window size* $k = 2$*, the set of accepting states* $Q_A = \{q_A\}$*, the set of rejecting states* $Q_R = \{q_R\}$ *and the transition function* $\delta(q_0,¢\$) = \{(q_A,HALT)\},\ \delta(q_0,aa) = \{(q_1,PREPARE)\}$, $\delta(q_0,¢a) = \{(q_0,MVR)\},\quad \delta(q_1,aa) = \{RESTART(\lambda)\}$, $\delta(q_0,a\$) = \{(q_R,HALT)\}$.
The automaton has only one restarting instruction $\iota = \delta(q_1,aa) \to_\delta RESTART(\lambda)$*. At first glance this instruction seems to be pumping as it can be applied iteratively. The rewritten word* $w = aa$ *can be written as* u_1vu_2*, where either* $u_1 = aa$*,* $v = \lambda$*, and* $u_2 = \lambda$*, or* $u_1 = a$*,* $v = \lambda$*, and* $u_2 = a$*. If* $u_1 = aa$ *and* $xu_1vu_2y = xaay \Rightarrow_\iota xy = xvy$ *for some* $x,y \in \{a\}^*$*, then it does not hold* $xu_1^{j-1}\underline{u_1vu_2}u_2^{j-1}y \Rightarrow_\iota xu_1^{j-1}vu_2^{j-1}y$*, for all* $j \geq 1$*, as the deleting realized by the corresponding PREPARE and RESTART instructions is performed at the beginning of the word and not around v, for* $j > 1$.

Similarly, when $u_1 = a$*,* $v = \lambda$*, and* $u_2 = a$*, it can be shown that the condition* $xu_1^{j-1}\underline{u_1vu_2}u_2^{j-1}y \Rightarrow_\iota xu_1^{j-1}vu_2^{j-1}y$*, for all* $j \geq 1$*, does not hold.*

To obtain a pumping RP*-automaton accepting the same language, we can change the automaton a bit so that it realizes the RESTART operation at the end of the word. For that we increase the window size to 3 and obtain* RP*-automaton* $N_2 = (Q,\Sigma,¢,\$,q_0,k,\delta,Q_A,Q_R)$ *with the set of states* $Q = \{q_0,q_1,q_A,q_R\}$*, the alphabet* $\Sigma = \{a\}$*, window size* $k = 3$*, the set of accepting states* $Q_A = \{q_A\}$*, the set of rejecting states* $Q_R = \{q_R\}$ *and the transition function* $\delta(q_0,¢\$) = \{(q_A,HALT)\},\ \delta(q_0,¢a\$) = \{(q_R,HALT)\}$, $\delta(q_0,aaa) = \{(q_0,MVR)\},\ \delta(q_0,aa\$) = \{(q_1,PREPARE)\}$, $\delta(q_0,¢aa) = \{(q_0,MVR)\},\ \delta(q_1,aa) = \{RESTART(\lambda)\}$.
It is easy to see that the new automaton N_2 *is pumping as it has single one-side pumping instruction that can be applied only at the right end of its tape.*

Note that N_2 *is in the strong cycling form.*

The next lemma follows from the correctness preserving property.

Lemma 1. *Let* $M = (Q,\Sigma,¢,\$,q_0,k,\delta,Q_A,Q_R)$ *be a det-mon-*RP*-automaton of window size k, let* $p = |Q|$*, and* $\iota = (q_\iota,w) \to_\delta RESTART(v)$ *be a restarting instruction of M. The instruction* ι *is a pumping instruction of M with pumping words* u_1,u_2 *iff for all* $x,y \in \Sigma^*$ *such that* $xu_1vu_2y \Rightarrow_\iota xvy$ *and each j,* $1 \leq j \leq p+k+1$ *it holds that*

$$xu_1^{j-1}\underline{u_1vu_2}u_2^{j-1}y \Rightarrow_\iota xu_1^{j-1}vu_2^{j-1}y. \qquad (1)$$

Proof: Obviously, if ι is a pumping instruction then for all $x,y \in \Sigma^*$ such that $xu_1vu_2y \Rightarrow_\iota xvy$ the condition (1) holds for each $j \geq 1$.

To prove the opposite implication, let x,y be words such that $xu_1vu_2y \Rightarrow_\iota xvy$ and for each j, $1 \leq j \leq p+k+1$ the condition (1) holds. We will show that the condition (1) is true also for any $j > p+k+1$. Let $w(j) = xu_1^jvu_2^jy$, for all $j \geq 0$. From (1) it follows that for each j, $1 \leq j \leq p+k+1$, the computation on $w(j)$ proceeds as follows:

$$\begin{aligned} q_0¢w(j)\$ = q_0¢xu_1^jvu_2^jy\$ &\vdash_M^* ¢xq_{j_1}u_1^jvu_2^jy\$ \\ &\vdash_M^* ¢xu_1q_{j_2}u^{j-1}vu_2^jy\$ \\ &\vdash_M^* \ldots \\ &\vdash_M^* ¢xu^{j-1}q_{j_j}u_1vu_2^jy\$ \\ &\vdash_M^* ¢xu^{j-1}q_\iota u_1vu_2^jy\$ \\ &\vdash_M^* q_0¢xu^{j-1}vu_2^{j-1}y\$ \\ &\vdash_M^* \ldots \end{aligned}$$

where q_{j_i} denotes the state in which the i-th copy of u_1 is the prefix of the read/write window contents in the first cycle of the computation of M on $w(j)$ and $(q_{j_j},u_1vu_2\omega) \to_\delta (q_\iota,PREPARE)$ is the preparing instruction followed by the instruction ι in the cycle. Further, let C_{j_i} denote the configuration of M corresponding to q_{j_i} during the first cycle on the word $w(j)$.

Consider the first cycle of the computation of M on $w(j)$, where $j > p+k+1$. While the whole read/write window of M is inside the prefix xu_1^{p+k+1} of $w(j)$, M executes the same instructions as in the first cycle on $w(p+k+1)$. Moreover, the contents of the read/write window of M is the same in all configurations C_{j_i} for all i, $1 \leq i \leq p+1$, because $|u_1^k| \geq k$. Since M has p states, there are two positive integers r,d, $1 \leq r < r+d \leq p+1$ such that $q_{j_r} = q_{j_{r+d}}$ and M executes from the configuration $C_{j_{r+d}}$ the same sequence of instruction as between the configurations C_{j_r} and $C_{j_{r+d}}$ (on $w(j)$, where $j \geq r+2d+k$).

Now, we can prove that (1) holds for any $j \geq 1$. The base statement that the condition (1) holds for each j, $1 \leq j \leq p+k+1$ is trivially satisfied. Let us suppose that (1) holds for each j, $1 \leq j \leq n$, where $n \geq p+k+1$. We will show that (1) holds also for $j = n+1$. During the first cycle on $w(n+1)$, the automaton executes between the configurations $C_{(n+1)_{r+d}}$ and $C_{(n+1)_{n+1}}$ on the word $w(n+1)$ exactly the same sequence of instructions as between the configurations $C_{(n+1-d)_r}$ and $C_{(n+1-d)_{(n+1-d)}}$ on the word $w(n+1-d)$. Therefore, it holds $w(n+1) = xu_1^n\underline{u_1vu_2}u_2^ny \Rightarrow_\iota xu_1^nvu_2^ny$ and together with the assumption of the induction step, it holds $xu_1^{j-1}\underline{u_1vu_2}u_2^{j-1}y$, for all j, $1 \leq j \leq n+1$. This completes the proof of the lemma. □

Note that Lemma 1 could be used for testing, whether a given restarting instruction is pumping, if we were able to bound the length of x,y in it. As a corollary of the following proposition and lemma we get that it is the case.

Proposition 3. *For any deterministic* RP*-automaton M of window size k, there is a deterministic* RP*-automaton* M' *of window size n,* $n \geq k$*, such that* M' *is in the strong cyclic form and* $L(M) = L(M')$*. In addition, when M is monotone then* M' *is monotone as well, and when M is pumping then* M' *is pumping as well, and if* $u \Rightarrow_M v$ *then* $u \Rightarrow_{M'} v$.

Proof: An RP-automaton is in the strong cyclic form if it only accepts and rejects in tail computations words of bounded length. Thus, if the original RP-automaton was

allowed to accept/reject in a tail computation on a word longer than the window size, we have to force it to perform one or more cycles so that in a tail computation it finally accepts (rejects) a word that together with the end-markers fits into the read/write window.

We can assume that M always accepts or rejects in a configuration in which it scans the right sentinel \$. Otherwise, we can modify it so that instead of an "original" accepting (rejecting) state, it would enter a special state that causes moving to the right end and then accepting (rejecting). Since the language of words accepted (rejected) in a tail computation is regular, there are finite automata A and A^{C} accepting these languages.

A new RP-automaton M' will be of window size $k' = \max\{n_A+1, n_C+1, n_M\}$, where n_A, n_C and n_M are the numbers of states of A, A^{C} and M, respectively. When moving right, the automaton M' simultaneously simulates the computations of A, A^C and M. The pumping lemma for regular languages implies that if M accepts or rejects w of length greater than k' in a tail computation, then $w \in L(A)$ or $w \in L(A^{\mathrm{C}})$ and for some words x,y,z it holds: $w = xyz$, $|yz| < k'$, $|y| > 0$ and $xz \in L(A)$ or $xz \in L(A^{\mathrm{C}})$.

The above modification ensures that when M accepts or rejects, it has already read the whole tape till the right sentinel. If M would accept or reject but M' does not have also the left sentinel ¢ in its read/write window, the automaton M', instead of accepting/rejecting, deletes y by applying a suitable pair of preparing and restarting instruction of the form $(q_{yz}, yz\$) \to_\delta (q_{Iyz}, PREPARE)$ and $(q_{Iyz}, y) \to RESTART(\lambda)$, for some new states q_{yz}, q_{Iyz}. Obviously, M' is in strong cyclic form, $L(M') = L(M)$ and $u \Rightarrow_M v$ implies $u \Rightarrow_{M'} v$.

The described simulation preserves monotonicity and also pumping property, because all added restart operations are pumping and performed at the right end of the tape. □

The next lemma enables extending Lemma 1 to det-mon-RP-automata in the strong cyclic form.

Lemma 2. *Let $M = (Q, \Sigma, ¢, \$, q_0, k, \delta, Q_{\mathrm{A}}, Q_{\mathrm{R}})$ be a det-mon-*RP*-automaton with window size k, let $p = |Q|$, and let $\iota = (q_\iota, w) \to_\delta RESTART(v)$ be a restarting instruction of M. There exists a constant m such that ι is a pumping instruction of M with pumping words u_1, u_2 iff for all $x, y \in \Sigma^*$ satisfying $|x| \le m$, $|y| < k$, $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$ it holds*

$$\begin{aligned} &\text{for each } j, 1 \le j \le p+k+1 : \\ &\qquad x u_1^{j-1} \underline{u_1 v u_2} u_2^{j-1} y \Rightarrow_\iota x u_1^{j-1} v u_2^{j-1} y. \end{aligned} \tag{2}$$

Proof: Obviously, if ι is a pumping instruction then condition (2) is met for all words x, y such that $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$ and for each $j \ge 1$.

Let $m = d + k + 1$, where d denotes the number of possible pairwise different instructions of M. Evidently, $d = p \cdot |\mathscr{PC}^{(k)}| \le p \cdot (|\Sigma|+3)^k$. We will show that if condition (2) is satisfied for all $x, y \in \Sigma^*$ such that $|x| \le m$, $y < k$, $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$, then condition (2) is satisfied also for any $x, y \in \Sigma^*$ of arbitrary length such that $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$. Then, by applying Lemma 1, we obtain that ι is a pumping instruction. The proof will be split into two claims.

Claim 1. *Assume that condition (2) holds for all words x, y, $|x| \le m$, $|y| < k$ such that $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$. Then condition (2) holds for all words x, y, $|x| \le m$ and y of arbitrary length, such that $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$.*

Proof of claim: According to the assumptions of the claim, for y of length at most $k-1$ the condition (2) holds trivially. If $|y| \ge k$ and $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$ then during the corresponding cycle the automaton M visited at most $k-1$ symbols to the right from $u_1 v u_2$, i.e. at most the first $k-1$ symbols of y. Therefore, $x\underline{u_1 v u_2}y' \Rightarrow_\iota xvy'$ is true for the prefix y' of y such that $y = y'y''$ and $|y'| = k-1$ for some word y''. Then, according to the assumption of the claim, it holds $xu_1^{j-1}\underline{u_1 v u_2}u_2^{j-1}y' \Rightarrow_\iota xu_1^{j-1}vu_2^{j-1}y'$ and also $xu_1^{j-1}\underline{u_1 v u_2}u_2^{j-1}y'y'' \Rightarrow_\iota xu_1^{j-1}vu_2^{j-1}y'y''$, for all j, $1 \le j \le p+k+1$, as no symbol of y'' was visited in any of the corresponding cycles. Hence, the condition (2) holds when we do not restrict the length of y. □

Claim 2. *Assume that condition (2) holds for all words x, y, $|x| \le m$, y of arbitrary length, such that $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$. Then the condition (2) holds for all words x and y of arbitrary lengths, such that $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$.*

Proof of claim: We will show that the condition (2) holds for arbitrary x by induction on the length of x.

Induction basis: For x of length at most m the condition (2) is true trivially.

Induction step: Let the claim be true for x of length at most n for some $n \ge m$. We will show that the condition (2) is true also for x of length $n+1$. Let x be a word of length $n+1$.

In the first cycle of M on the word xu_1vu_2y, more than d *MVR* steps were performed while the read/write window was completely inside the word x. At least two of them were according to the same *MVR* instruction in a state q. Hence, we can write $x = x_1x_2x_3$, for some words x_1, x_2, x_3 such that $|x_2| > 0$, $|x_3| \ge k$ and

$$\begin{aligned} q_0¢x_1x_2x_3u_1vu_2y\$ &\vdash_M^* ¢x_1qx_2x_3u_1vu_2y\$ \\ &\vdash_M^* ¢x_1x_2qx_3u_1vu_2y\$ \\ &\vdash_M^* ¢x_1x_2x_3q_\iota u_1vu_2y\$ \\ &\vdash_M^* q_0¢x_1x_2x_3vy\$, \end{aligned}$$

where the prefix of length k of x_2x_3 is the same as the prefix of length k of x_3. That is, in both above configurations with the state q the automaton executes the same *MVR* instruction.

Therefore, when we leave out x_2 and the corresponding steps of the cycle, we obtain a valid cycle of M on the shorter word $x_1x_3u_1vu_2y$. Thus, it holds $x_1x_3\underline{u_1 v u_2}y \Rightarrow_\iota x_1x_3vy$ and we can apply the assumption of the induction step that the condition (2) holds for words x of length at

most n for the word x_1x_3 to obtain that

$$\text{for each } j, 1 \leq j \leq p+k+1 : \quad x_1x_3u_1^{j-1}\underline{u_1vu_2}u_2^{j-1}y \Rightarrow_\iota x_1x_3u_1^{j-1}vu_2^{j-1}y.$$

Obviously, in each of the corresponding cycles we can insert back x_2 between x_1 and x_3 and the corresponding sequence of steps to obtain that

$$x_1x_2x_3u_1^{j-1}\underline{u_1vu_2}u_2^{j-1}y \Rightarrow_\iota x_1x_2x_3u_1^{j-1}vu_2^{j-1}y$$

is true for all j, $1 \leq j \leq p+k+1$, which completes the proof of the claim. □

By combining Claim 2 and Lemma 1, it follows that the instruction ι is pumping. □

Corollary 1. *Let M be a scf-det-mon-*RP*-automaton, and ι a restarting instruction of M. It is decidable whether ι is a pumping instruction or not.*

The characterization of DCFL by R-automata was given in [3].

Theorem 2 ([3])**.** DCFL $= \mathscr{L}$(det-mon-R).

We show that DCFL can even be characterized by pmp-RP-automata.

Theorem 3. DCFL $= \mathscr{L}$(pmp-RP) $= \mathscr{L}$(det-mon-RP)

Proof: The theorem is a consequence of the following two lemmas.

Lemma 3. $\mathscr{L}$(det-mon-RP) $\subseteq$ DCFL.

Proof: As the models of det-mon-R- and det-mon-RP-automata differ only slightly, we can use here a slightly modified proof of Lemma 8 in [3] stating that $\mathscr{L}$(det-mon-R) $\subseteq$ DCFL. For a given det-mon-RP-automaton M, a method from [3] can be used to construct a deterministic push-down automaton P that accepts the same language as M. □

To show the opposite direction, we use the characterization of deterministic context-free languages by means of $LR(1)$-grammars in Greibach Normal Form and $LR(1)$-analyzers (cf., e.g., [1])[1].

Lemma 4. DCFL $\subseteq \mathscr{L}$(pmp-RP).

Proof: The inclusion follows from an analysis of the det-mon-R-automaton M simulating a syntactic analysis of a DCFL language L in [3]. Each det-mon-R-automaton M can be easily converted into a det-mon-RP-automaton M' by splitting each restarting instruction of M into one preparing instruction and one restarting instruction of M'. To see, that the resulting det-mon-RP-automaton M' is

[1] Recall that context-free grammar $G = (N, T, P, S)$ is LR(k) for $k \geq 0$ if for any string α exists unique partition $\alpha = \beta\gamma u$, where $\alpha, \beta, \gamma \in (N \cup T)^*$, $u \in T^*$, such that there is a rightmost derivation $S \Rightarrow_r^* \beta Au \Rightarrow_r \alpha\beta u$ where A is a nonterminal.

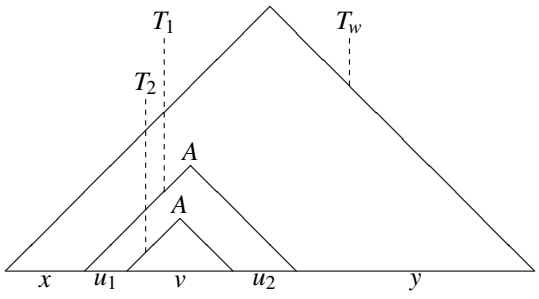

Figure 1: The structure of a derivation tree.

pumping we sketch the construction of M; the construction will also be helpful for better understanding of later results and proofs.

It is well known that if L' is a deterministic context-free language and \$ is a symbol not in the alphabet of L', then $L' \cdot \{\$\}$ is a deterministic prefix-free context-free language that can be parsed by a LR(0)-analyzer P_0. This fact was used in [3]. Here, in order to construct a pmp-det-mon-RP-automaton, we use a similar construction of an LR(1)-analyzer P of L' that is based on an LR(1)-grammar in Greibach Normal Form. The existence of such analyzer for any DCFL is proved in [1].

Based on the simulation of P on a word w we can construct the derivation tree T_w (the inner vertices of which are labeled with nonterminals and leaves correspond to terminal symbols). Thus, for any word $w \in L$ there is exactly one derivation tree T_w. The standard pumping lemma for context-free languages implies existence of two constants $p, q > 0$ such that for any word w with length greater than p there are (complete) subtrees T_1 and T_2 of T_w such that T_2 is a subtree of T_1 and roots of both subtrees have the same label (cf. Fig. 1); in addition, T_2 has fewer leaves than T_1, T_1 has at most q leaves and $|u_1| > 0$. The word u_1 is nonempty, because the right-hand side of the rule used to rewrite the nonterminal A in the root of T_1 must start with a terminal (the grammar is in Greibach Normal form).

Obviously, replacing T_1 with T_2, we get the derivation tree $T_{w(0)}$ for a shorter word $w(0)$ (if $w = xu_1vu_2y$ then $w(0) = xvy$). Analogously, replacing T_2 with T_1, we get the derivation tree $T_{w(2)}$ for a longer word $w(2)$ where $w(2) = xu_1^2vu_2^2y$. If we repeat this replacing of T_2 with T_1 i-times we obtain the derivation tree $T_{w(i+1)}$ for a word $w(i+1)$ where $w(i+1) = xu_1^{i+1}vu_2^{i+1}y$.

The key to a construction of the det-mon-R-automaton M in [3] is the possibility to identify the leftmost subword u_1vu_2 corresponding to subtrees T_1 and T_2 as shown in Fig. 1 reading from left to right with the help of constant size memory only. In its constant size memory M stores all maximal subtrees of the derivation tree with all their leaves in the buffer. When it identifies a subtree like T_1 above, M performs the corresponding deleting by a *RESTART* operation. Obviously, the read/write window of length $k > q$ is sufficient for that.

If that is not the case then M forgets the leftmost of these subtrees with all its $n \geq 1$ leaves, and reads n new symbols to the right end of the buffer (performing *MVR*-instructions). Then M continues constructing the maximal subtrees with all leaves in the (updated) buffer (simulating P). Short words of length less than k are accepted/rejected in tail computations.

To obtain M' it is sufficient to use instead of restarting instructions of M the corresponding preparing and restarting instructions of RP-automaton. The preparing and restarting instructions can handle LR(1)-analysis in the same way as M the LR(0)-analysis. The resulting RP-automaton M' preserves the determinism and monotonicity of M; from the construction it is clear that M' is pumping as well. □

Note. While the previous proof is based on the paper [3], a similar proof can be based on the constructions from paper [7] based on deterministic list automata.

The following corollary is a consequence of the previous theorem and Proposition 3.

Notation. In what follows, we will write pmsc- instead of pmp-scf-.

Corollary 2.

$$\text{DCFL} = \mathscr{L}(\text{pmsc-RP}) = \mathscr{L}(\text{scf-det-mon-RP}).$$

In what follows, we aim to obtain some conditions for a det-RP-automaton to accept a non-regular language. At first, we conjecture, that each pmp-det-mon-RP-automaton that does not have any two-side pumping instruction generates a regular language. On the other hand, a pmp-det-mon-RP-automaton having a two-side pumping instruction can still accept a regular language. This can be seen in the following example.

Example 3. *Let* $M_3 = (Q, \Sigma, ¢, \$, q_0, k, \delta, Q_A, Q_R)$ *be the* RP-*automaton with the set of states* $Q = \{q_0, q_1, q_a, q_b, q_A\}$*, the alphabet* $\Sigma = \{a, b\}$*, window size* $k = 2$*, the set of accepting states* $Q_A = \{q_A\}$*, the set of rejecting states* $Q_R = \emptyset$ *and the transition function*

$\delta(q_0, ¢\$) = \{(q_A, HALT)\}$, $\delta(q_0, bb) = \{(q_0, MVR)\}$,
$\delta(q_0, ¢a) = \{(q_0, MVR)\}$, $\delta(q_0, ab) = \{(q_1, PREPARE)\}$,
$\delta(q_0, ¢b) = \{(q_0, MVR)\}$, $\delta(q_1, ab) = \{RESTART(\lambda)\}$,
$\delta(q_0, aa) = \{(q_0, MVR)\}$, $\delta(q_0, a\$) = \{(q_a, PREPARE)\}$,
$\delta(q_0, ba) = \{(q_0, MVR)\}$, $\delta(q_0, b\$) = \{(q_b, PREPARE)\}$
$\delta(q_a, a) = \{RESTART(\lambda)\}$, $\delta(q_b, b) = \{RESTART(\lambda)\}$.

The automaton differs from the automaton M_1 *of Example 1 only slightly. It has two new restarting states* q_a *and* q_b *that enable to delete any symbol to the left from the right sentinel \$. Nevertheless, the first restarting instruction* $\iota = (q_1, ab) \to_\delta RESTART(\lambda)$ *is still two-side pumping instruction (see Example 1). The automaton* M_3 *is deterministic and monotone. In contrast to* M_1*, it is in the strong cyclic form and accepts all words over the alphabet* $\{a, b\}$.

Hence, we define a property of two-side pumping instructions which ensures that the resulting det-mon-RP-automaton does accept a non-regular language.

Definition 4. *Let* $M = (Q, \Sigma, ¢, \$, q_0, k, \delta, Q_A, Q_R)$ *be a* pmp-RP-*automaton accepting the language* $L = L(M)$*. Let* $\iota = (q_r, u_1 v u_2) \to_\delta RESTART(v)$ *be a two-side pumping restarting instruction of* M *with pumping words (strings)* u_1, u_2 *and* $x\underline{u_1 v u_2}y \Rightarrow_\iota xvy$*, for some* $x, y \in \Sigma^*$.

Let p *be a positive integer. We say that* ι *is a* (p, x, y)-distinguishing instruction *for* M *if at least one of the following cases occurs:*

(I) $xu_1^m v u_2^m y \in L$*, and* $xu_1^m v u_2^m u_2^{p \cdot j} y \notin L$*, for all* $j \geq 1$*,* $m \geq 0$*,*

(II) $xu_1^m v u_2^m y \in L$*, and* $xu_1^{p \cdot j} u_1^m v u_2^m y \notin L$*, for all* $j \geq 1$*,* $m \geq 0$*,*

(III) $xu_1^m v u_2^m y \notin L$*, and* $xu_1^m v u_2^m u_2^{p \cdot j} y \in L$*, for all* $j \geq 1$*,* $m \geq 0$*,*

(IV) $xu_1^m v u_2^m y \notin L$*, and* $xu_1^{p \cdot j} u_1^m v u_2^m y \in L$*, for all* $j \geq 1$*,* $m \geq 0$.

We say that ι *is a distinguishing instruction for* M *if there are* p, x, y *such that* ι *is a* (p, x, y)*-distinguishing instruction for* M.

We say that M *is a distinguishing* RP-*automaton if there is a distinguishing instruction* ι *for* M*. We write dist-RP-automaton to denote a distinguishing pmp-RP-automaton.*

Example 4. *Let us consider the automaton* $M_1 = (Q, \Sigma, ¢, \$, q_0, k, \delta, Q_A, Q_R)$ *from Example 1. We will show that* M_1 *is a* dist-RP-*automaton. The automaton has only one restarting instruction* $\iota = (q_1, ab) \to_\delta RESTART(\lambda)$*. This instruction is a two-side pumping instruction with pumping strings* $u_1 = a$ *and* $u_2 = b$ *and* $\underline{ab} = x\underline{u_1 v u_2}y \Rightarrow_1 xvy = \lambda$*, for* $x = \lambda$*,* $v = \lambda$*,* $y = \lambda$*. The instruction* ι *is a* (p, x, y)*-distinguishing instruction for* M_1 *for* $p = 1$*. Namely, for all* $j \geq 1, m \geq 0$ *it holds*

$$xu_1^m v u_2^m y = a^m b^m \in L(M_1) \text{ and}$$
$$xu_1^m v u_2^m u_2^{p \cdot j} y = a^m b^{m+j} \notin L(M_1),$$

and also

$$xu_1^m v u_2^m y = a^m b^m \in L(M_1) \text{ and}$$
$$xu_1^m u_1^{p \cdot j} v u_2^m y = a^{m+j} b^m \notin L(M_1).$$

Theorem 4. *Let* $L = L(M) \neq \emptyset$ *be a deterministic context-free language accepted by a* dist-RP-*automaton* M*. Then* L *is a non-regular language.*

Proof. To obtain a contradiction, we suppose that $M = (Q, \Sigma, ¢, \$, q_0, k, \delta, Q_A, Q_R)$ is a dist-RP-automaton accepting a regular language $L = L(M)$. Since M is a dist-RP-automaton, it has a (p, x, y)-distinguishing instruction $\iota = (q_\iota, u_1 v u_2) \to_\delta RESTART(v)$, for some $x, y, v \in \Sigma^*$, $u_1, u_2 \in \Sigma^+$, $q_\iota \in Q$ and $p \geq 1$. As L is regular, there exists a deterministic finite automaton A with n_A states accepting the language $L(A) = L$.

The proof follows by the analysis of the four possible cases of (p, x, y)-distinguishing property of ι:

Case (I): From the definition, for all $m \geq 0$, $j \geq 1$ it holds $xu_1^m v u_2^m y \in L$ and $xu_1^m v u_2^m u_2^{p \cdot j} y \notin L$. Let q_{m_j} denote the

state of the automaton A in which it reads the first symbol of the j-th copy of u_2 to the right from v. That is, A is in the state q_{m_j} after reading $xu_1^m vu_2^{j-1}$, for $j \geq 1$. For $m > n_A$, there exist integers r, s, $1 \leq r < s \leq n_A + 1$ such that $q_{m_r} = q_{m_s}$. Then, for all $i \geq 0$, it holds $q_{m_r} = q_{m_{r+i\cdot(r-s)}}$ and the automaton A accepts all words of the form $xu_1^m vu_2^{m+i\cdot(s-r)}y$. For $i = p$, we obtain that $xu_1^m vu_2^{m+p\cdot(s-r)}y \in L$ which contradicts the assumption $xu_1^m vu_2^m u_2^{p\cdot j}y \notin L$, for $j = (s-r)$.
Case (II): From the definition, for all $m \geq 0$, $j \geq 1$ it holds $xu_1^m vu_2^m y \in L$, and $xu_1^{p\cdot j}u_1^m vu_2^m y \notin L$. Let q_{m_j} denote the state of the automaton A in which it reads the first symbol of the j-th copy of u_1 to the right from x. That is, A is in the state q_{m_j} after reading xu_1^{j-1}, for $j \geq 1$. For $m > n_A$, there exist integers r, s, $1 \leq r < s \leq n_A + 1$ such that $q_{m_r} = q_{m_s}$. Then, for all $i \geq 0$, it holds $q_{m_r} = q_{m_{r+i\cdot(r-s)}}$ and the automaton A accepts all words of the form $xu_1^{m+i\cdot(s-r)}vu_2^m y$. For $i = p$, we obtain that $xu_1^{m+p\cdot(s-r)}vu_2^m y \in L$ which contradicts the assumption $xu_1^{p\cdot j}u_1^m vu_2^m y \notin L$, for $j = (s-r)$.

Analysis of conditions (III) and (IV) from Definition 4 is analogous to the shown cases. □

As each dist-RP-automaton is deterministic and monotone (Definition 4), it accepts a deterministic context-free language (Lemma 3). Hence, Lemma 3, Theorem 4 and Proposition 3 imply the following corollary ($\subset$ denotes the proper subset relation).

Corollary 3. $\mathscr{L}(\text{dist-RP}) = \mathscr{L}(\text{scf-dist-RP}) \subset \text{DCFL}$.

Remark. We conjecture that $\mathscr{L}(\text{scf-dist-RP})$ is equal to the set of all non-regular DCFL. But it remains an open problem.

4 Conclusions

We plan in the near future to show that any pmsc-RP-automaton M which accepts a non-regular language can be transformed in a scf-dist-RP-automaton, for which all its two-side pumping instructions are distinguishing. Let us denote such type of automata as strongly distinguishing RP-automata (sdist-RP-automata). The sdist-RP-automata will allow to extend the results from [8] achieved by deterministic context-free grammars. Further, it will allow to introduce some types of degrees of non-regularity of DCFL, e.g., according to the number of distinguishing instructions in a strongly distinguishing RP-automaton. The combinations of this measure with other measures typical for restarting automata (e.g., the length of rewriting windows) will give us natural measures for complexity of DCFL.

Finally, sdist-RP-automata will create a nice tool for localization and measures for syntactic errors in deterministic context-free languages.

5 Acknowledgements

We thank the anonymous referees whose comments and suggestions have helped to improve the presentation of this paper.

References

[1] M. M. Geller, M. A. Harrison, I. M. Havel: Normal forms of deterministic grammars. Discrete Mathematics, 16(4):313–321 (1976)

[2] J. Hopcroft, J. Ullman: *Introduction to Automata Theory, Languages, and Computation*; Addison-Wesley (1979)

[3] P. Jančar, F. Mráz, M. Plátek, J. Vogel: *Restarting Automata*, Proceedings of FCT 1995, LNCS 965, Springer, 283–292 (1995)

[4] P. Jančar, F. Mráz, M. Plátek, and J. Vogel. On restarting automata with rewriting. In G. Păun and A. Salomaa, editors, *New Trends in Formal Language Theory (Control, Cooperation and Combinatorics)*, volume 1218 of *LNCS*, pages 119–136. Springer, 1997.

[5] P. Jančar, F. Mráz, M. Plátek, J. Vogel: On Monotonic Automata with a Restart Operation. Journal of Automata, Languages and Combinatorics 4(4): 287–311 (1999)

[6] F. Otto: Restarting Automata. Recent Advances in Formal Languages and Applications; Studies in Computational Intelligence, vol 25, Springer, 269–303 (2006)

[7] M. Plátek, J. Vogel: Deterministic list automata and erasing graphs. The Prague bulletin of mathematical linguistics 45, 27–50 (1986)

[8] J. Šíma, M. Plátek: One Analog Neuron Cannot Recognize Deterministic Context-Free Languages. Proceedings of ICONIP 2019, Part III, LNCS 11955, 77–89, Springer (2019)

[9] J. Šíma, M. Plátek: The Simplest Non-Regular Deterministic Context-Free Languages. *In preparation*.

[10] S. Sippu, E. Soisalon-Soininen: Parsing Theory, Volume II: LR(k) and LL(k) Parsing. Monographs in Theoretical Computer Science, Volume 20 of EATCS Series, Springer (1990)