Pumping Deterministic Monotone Restarting Automata and DCFL

František Mráz ¹, Dana Pardubská ², Martin Plátek ¹,³ and Jiří Šíma ¹

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
pardubskadcs.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 $\epsilon$ 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 $\epsilon$ 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, $M = (Q, \Sigma, \epsilon, S, q_0, k, \delta, Q_A, Q_R)$ (with $k$-bounded lookahead) is a device with a finite state control unit with the
finite set of states $Q$ containing two disjoint 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 $\epsilon$, the last one the right sentinel symbol $\$, and each other item contains a symbol from a finite alphabet $\Sigma$ (not containing $\epsilon$, $\$). The head has a flexible read/write window of length at most $k$ (for some $k$ in $Q - M$ 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 $\epsilon$ and $\$, the control unit is in the initial state $q_0$, and the window scans the left sentinel $\epsilon$ 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

$$\delta : (Q \setminus (Q_A \cup Q_R)) \times P(\leq k) \rightarrow P(Q \times \{MVR, PREPARE\}) \cup \{RESTART(v) \mid v \in \leq (k-1)\}.$$ 

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

$$P(i) := \{(\epsilon \cdot \Sigma^{i-1}) \cup \Sigma^i \cup \Sigma^i \cdot \{\}\} \cup \{(\epsilon \cdot \Sigma^{j-2}) \cdot \{\}\},$$

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

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

1. A move-right instruction of the form $(q, u, v) \rightarrow_{q'} (q', MVR)$ 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 $(q, u, v) \rightarrow_{q''} (q'', PREPARE)$ changes $M$'s state to a restarting state $q''$ that determines the next instruction, which must be a restarting instruction.

3. A restarting instruction $I$ is of the form $(q_I, w) \rightarrow_{q_I} \text{RESTART}(v)$, 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 $\epsilon$. 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 $(q, u) \rightarrow_{q'} (q', \text{HALT})$, where $q' \in Q_A$ or $q' \in Q_R$, 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 $Q_A \cup Q_R$, 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 $\alpha \epsilon \beta$, where $q \in Q$, and either $\alpha = \lambda$ and $\beta \in \{\epsilon \cdot \Sigma^* \cdot \{\}\}$ or $\alpha \in \{\epsilon \cdot \Sigma^* \}$ 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 $\beta$ or all symbols of $\beta$ if $|\beta| < k$. An initial (restarting) configuration is of the form $q_0 \epsilon w \$, where $w \in \Sigma^*$. A rewriting configuration is of the form $\alpha \epsilon \beta$, where $q_I$ is a restarting state.

A computation of $M$ is a sequence $C = C_0, C_1, \ldots, C_I$ of configurations of $M$, where $C_I$ is a restarting configuration and $C_{I+1}$ is obtained from $C_I$ by a step of $M$, for all $\ell$, $0 \leq \ell < j$, denoted as $C_I \vdash M C_{I+1}$ 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 $\epsilon$, $\$) 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 lucidity 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 end-marker \( \$ \).

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. \( \mathcal{R} \) denotes the class of all (non-deterministic) RP-automata. Prefix der- denotes the deterministic version, similarly mon- the monotone version. Prefix scf- denotes the version in the strong cyclic form. \( \mathcal{L}(A) \), 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 \( \mathcal{L}(\text{det-mon-RP}) \).

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

**Proposition 2.** The classes \( \mathcal{L}(\text{det-mon-RP}) \) and \( \mathcal{L}(\text{det-RP}) \) 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 \( M \) 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 \( \text{RESTART} \) operation of RW-automaton is unambiguously split into two consecutive instructions \( \text{PREPARE} \) and \( \text{RESTART} \). This splitting has no influence on the decidability proof, since whenever the \( \text{RESTART} \) operation of RW-automaton is used/simulated in the original proof it can unambiguously be replaced by \( \text{PREPARE} \) and \( \text{RESTART} \) operation of RP-automaton. □

In what follows, we consider only deterministic RP-automata. We write \( \alpha \Rightarrow_{\beta} \gamma \) for words \( \alpha, \beta, \gamma \in \Sigma^* \) when \( \alpha \) was shortened to \( \beta \) in one cycle consisting of several \( \text{MVR} \) steps followed by a \( \text{PREPARE} \) instruction and the restarting instruction \( I = (q_i, w) \Rightarrow_{\delta} \text{RESTART}(v) \) with the restart state \( q_i \). Additionally, when \( \epsilon x w y \$ \) is the rewriting configuration corresponding to the reduction \( x w y \Rightarrow_t x y v \), for some words \( x, y, v, w \), we will underline the occurrence of \( w \) rewritten in the cycle. That is, we will write \( x w y \Rightarrow_t x y v \).

Analogously, \( \alpha \Rightarrow_{\beta} \gamma \) indicates a computation consisting of several such cycles that use the same starting restarting instruction \( I \) with the restarting state \( q_i \).

## 3 Pumping Restaring Automata

Let \( M = (Q, \Sigma, \epsilon, \delta, q_0, Q_A, Q_R) \) be a det-mon-RP-automaton, and \( I = (q_i, w) \Rightarrow_{\delta} \text{RESTART}(v) \) a restarting instruction of \( M \). We say that \( I \) is a pumping instruction of \( M \) if \( w \) can be written as \( w = u_1 v u_2 \), for some words \( u_1, u_2, u_1 \neq \lambda \), and, for all \( x, y, \in \Sigma^* \), \( x u_1 v u_2 y \Rightarrow_{\epsilon} x v u_2 y \) implies \( \epsilon u_1^{-1} u_2 \epsilon^{-1} v y \Rightarrow \epsilon u_1^{-1} v u_2^{-1} y \), for all \( j \geq 1 \).

We say that \( I \) has pumping words \( u_1, u_2 \). If \( u_2 \neq \lambda \) we say that \( I \) is a two-side pumping instruction. If \( u_2 = \lambda \) we say that \( I \) 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 \( \text{pmp-} \).

**Example 1.** Let \( M_1 = (Q, \Sigma, \epsilon, \delta, Q_A, Q_R) \) 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 \( \delta(q_0, \epsilon) = \{ (q_A, \text{HALT}) \} \), \( \delta(q_0, ab) = \{ (q_0, \text{MVR}) \} \), \( \delta(q_0, a) = \{ (q_0, \text{PREPARE}) \} \), \( \delta(q_0, c) = \{ (q_0, \text{MVR}) \} \), \( \delta(q_1, ab) = \{ (q_R, \text{HALT}) \} \), \( \delta(q_0, ba) = \{ (q_0, \text{MVR}) \} \), \( \delta(q_0, b) = \{ (q_R, \text{HALT}) \} \).

The automaton has only one restarting instruction \( I = (q_1, ab) \Rightarrow_{\delta} \text{RESTART}(\lambda) \). This instruction is a two-side pumping instruction as \( w = ab \) can be written as \( u_1 v u_2 \) where \( u_1 = a \) and \( u_2 = b \) are nonempty, \( v = \lambda \). If \( q_0 \epsilon x u_1 v u_2 y \$ = q_0 \epsilon x a b y \$ \Rightarrow_{\epsilon} q_0 \epsilon x q_1 v u_2 y \$ \Rightarrow_{\epsilon} q_0 \epsilon x y \$ \) for some \( x, y \in \{ a, b \}^* \), then it holds that \( x u_1^{-1} v u_2^{-1} y \Rightarrow_{\epsilon} x y \$ \Rightarrow_{\epsilon} x v u_2 y \$ \), for all \( j \geq 0 \), and \( x u_1^{-1} u_2 \epsilon^{-1} v y \Rightarrow_{\epsilon} x v u_2^{-1} y \), 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 $M_2$, $M_2 = (Q, \Sigma, \varepsilon, S, q_0, k, \delta, Q_a, Q_R)$ with the set of states $Q = \{q_0, q_1, q_a, q_\lambda\}$, the alphabet $\Sigma = \{a\}$, window size $k = 2$, the set of accepting states $Q_a = \{q_\lambda\}$, the set of rejecting states $Q_R = \{q_R\}$ and the transition function $\delta(q_0, x) = \{(q_a, HALT)\}$, $\delta(q_1, x) = \{(q_1, PREPARE)\}$, $\delta(q_a, x) = \{(q_0, MVR)\}$, $\delta(q_1, x) = \{(q_1, PREPARE)\}$, $\delta(q_a, \varepsilon) = \{(q_R, HALT)\}$. The automaton has only one restarting instruction $t = \delta(q_1, au) \rightarrow_\delta \text{RESTART}(\lambda)$. At first glance this instruction seems to be pumping as it can be applied iteratively. The rewritten word $w = aa$ can be written as $u_1v_2$ 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 $xu_1vy = xaxy \Rightarrow xy = xyy$ for some $x, y \in \{a\}$, then it does not hold $xu_1^{-1}u_1vyu_2^{-1}y \Rightarrow_1 xu_1^{-1}vyu_2^{-1}y$, 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 $xu_1^{-1}u_1vyu_2^{-1}y \Rightarrow_1 xu_1^{-1}vyu_2^{-1}y$, 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 $N_2 = (Q, \Sigma, x, S, q_0, k, \delta, Q_a, Q_R)$ with the set of states $Q = \{q_0, q_1, q_a, q_\lambda\}$, the alphabet $\Sigma = \{a\}$, window size $k = 3$, the set of accepting states $Q_a = \{q_\lambda\}$, the set of rejecting states $Q_R = \{q_R\}$ and the transition function $\delta(q_0, x) = \{(q_a, HALT)\}$, $\delta(q_1, x) = \{(q_R, HALT)\}$, $\delta(q_a, aa) = \{(q_0, MVR)\}$, $\delta(q_a, \varepsilon) = \{(q_1, PREPARE)\}$, $\delta(q_0, \varepsilon) = \{(q_0, MVR)\}$, $\delta(q_1, au) = \{(q_1, PREPARE)\}$, $\delta(q_a, \varepsilon) = \{(q_R, HALT)\}$. 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 $M = (Q, \Sigma, \varepsilon, S, q_0, k, \delta, Q_a, Q_R)$ be a deterministic RP-automaton of window size $k$, let $p = |Q|$, and $t = (q_1, w) \rightarrow_\delta \text{RESTART}(v)$ be a restarting instruction of $M$. The instruction $t$ is a pumping instruction of $M$ with pumping words $u_1, u_2$ iff for all $x, y \in \Sigma^*$ such that $xu_1vy \Rightarrow_1 xyv$ and each $j$, $1 \leq j \leq p + k + 1$ it holds that $xu_1^{-1}u_1vyu_2^{-1}y \Rightarrow_1 xu_1^{-1}vyu_2^{-1}y$.

**Proof:** Obviously, if $t$ is a pumping instruction then for all $x, y \in \Sigma^*$ such that $xu_1vy \Rightarrow_1 xyv$ the condition (1) holds for each $j \\geq 1$.

To prove the opposite implication, let $x, y$ be words such that $xu_1vy \Rightarrow_1 xyv$ 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 $w(j) = xu_1^{-1}vyu_2^{-1}y$ 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:

$$
q_0cw(j)S = q_0cxu_1^{-1}vyu_2^{-1}yS \
\vdash_M \exists q_j, u_1^{-1}vyu_2^{-1}yS \
\vdash_M \exists q_j, u_1^{-1}vyu_2^{-1}yS \
\vdash_M \ldots \
\vdash_M q_0cxu_1^{-1}vyu_2^{-1}yS \
\vdash_M \ldots
$$

where $q_j$ 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 $(q_j, u_1vyu_2) \rightarrow_\delta (q_1, PREPARE)$ is the preparing instruction followed by the instruction $t$ in the cycle. Further, let $C_j$ denote the configuration of $M$ corresponding to $q_j$ 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 $xu_1^{-1}u_2^{-1}y$ 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 $C_j$ for all $i, 1 \leq i \leq p + 1$, because $|x| \geq k$. Since $M$ has $p$ states, there are two positive integers $r, d$, $1 \leq r < r + d \leq p + 1$ such that $q_{r+i} = q_{r+d+i}$ and $M$ executes from the configuration $C_{r+i}$ the same sequence of instruction as between the configurations $C_r$ and $C_{r+d}$ on the word $w(j)$, 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 $j$, $1 \leq j \leq n$, where $n > p + k + 1$. 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 $C_{(n+1)+i}$ and $C_{(n+1)+d+i}$ on the word $w(n + 1)$ exactly the same sequence of instructions as between the configurations $C_{(n+1)+d-i}$ and $C_{(n+1)+d+(i-d)}$ on the word $w(n + 1 - d)$. Therefore, it holds $w(n + 1) = xu_1^{-1}u_2^{-1}yv$ and together with the assumption of the induction step, it holds $xu_1^{-1}u_1vyu_2^{-1}y$, 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 $w \notin L(A^c)$ and for some words $x, y, z$ it holds: $w = xyz$, $|y| > 0$ and $xz \in L(A)$ or $xz \in L(A^c)$.

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 $c$ 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 $(q_{zc}, yz) \rightarrow \delta(q_{tc}, \text{PREPARE})$ and $(q_{tc}, y) \rightarrow \text{RESTART}(q)$, for some new states $q_{tc}, q_{zc}$. 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 deterministic-RP-automata in the strong cyclic form.

**Lemma 2.** Let $M = (Q, \Sigma, \delta, Q_0, w_1, s, \delta, Q_A, Q_R)$ be a deterministic-RP-automaton with window size $k$, let $p = |Q|$, and let $t = (q_{tc}, w) \rightarrow \delta \text{RESTART}(v)$ be a restarting instruction of $M$. There exists a constant $m$ such that $t$ is a pumping instruction of $M$ with pumping words $u_1, u_2$ iff for all $x, y \in \Sigma^*$ satisfying $|x| \leq m$, $|y| < k$, $xu_1vy \Rightarrow_t xvy$ it holds

$$u_1^{j-1}u_2^{j-1}y \Rightarrow_t u_1^{j-1}v_1^{j-1}y$$

for each $j, 1 \leq j \leq p + k + 1$.

**Proof:** Obviously, if $t$ is a pumping instruction then condition 2 is met for all words $x, y$ such that $xu_1vy \Rightarrow_t xvy$ and for each $j > 1$.

Let $m = d + k + 1$, where $d$ denotes the number of possible pairwise different instructions of $M$. Evidently, $d = p \cdot \delta(|P|)$, where $|\Sigma| + 3$. We will show that if condition 2 is satisfied for all $x, y \in \Sigma^*$ such that $|x| \leq m$, $y < k$, $xu_1vy \Rightarrow_t xvy$, then condition 2 is satisfied also for any $x, y \in \Sigma^*$ of arbitrary length such that $xuyv \Rightarrow_t xvy$. Then, by applying Lemma 1 we obtain that $t$ is a pumping instruction. The proof will be split into two claims.

**Claim 1.** Assume that condition 2 holds for all words $x, y$, $|x| \leq m$, $|y| < k$ such that $xu_1vy \Rightarrow_t xvy$. Then condition 2 holds for all words $x, y$, $|x| \leq m$ and $y$ of arbitrary length, such that $xu_1vy \Rightarrow_t xvy$.

**Proof of claim:** According to the assumptions of the claim, for $y$ of length at most $k − 1$ the condition 2 holds trivially. If $|y| \geq k$ and $xu_1vy \Rightarrow_t xvy$ then during the corresponding cycle the automaton $M$ visited at most $k − 1$ symbols of the right from $u_1vy$, i.e., at most the first $k − 1$ symbols of $y$. Therefore, $xu_1vy \Rightarrow_t xvy$ 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 $xu_1y'vy'' \Rightarrow_t xu_1y''vy', y''$ for all $j, 1 \leq j \leq p + k + 1$, 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 $x, y$, $|x| \leq m$, $y$ of arbitrary length, such that $xu_1vy \Rightarrow_t xvy$. Then the condition 2 holds for all words $x$ and $y$ of arbitrary lengths, such that $xu_1vy \Rightarrow_t xvy$.

**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 $n \geq m$. 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 $xu_1vy$, 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 $|x_2| > 0$, $|x_3| \geq k$ and

$$q_0 \Box x_1x_2x_3u_1vy \Rightarrow_t x_1x_2x_3u_1vy$$

$$\vdash_{M} q_0 \Box x_1x_2x_3u_1vy$$

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 $x_1x_1u_1vy$. Thus, it holds $x_1x_1u_1vy \Rightarrow_t x_1x_1vy$ 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

$$x_1x_3u_i^{-1}w_1v_2u_2^{-1} \Rightarrow_i x_1x_3u_i^{-1}w_1v_2^{-1}y.$$  

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

$$x_1x_2x_3u_i^{-1}w_1v_2u_2^{-1} \Rightarrow_i x_1x_2x_3u_i^{-1}w_1v_2^{-1}y$$

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

By combining Claim[2] and Lemma[1] it follows that the instruction $i$ is pumping. \hfill \Box

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

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

**Theorem 2** ([3]), $\mathcal{L}(\text{det-mon-R})$.

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

**Theorem 3.** $\mathcal{L}(\text{pmp-RP}) = \mathcal{L}(\text{det-mon-RP})$

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

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

**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 2 in [3] stating that $\mathcal{L}(\text{det-mon-R}) \subseteq \mathcal{L}(\text{DCFL})$. 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 $M$. \hfill \Box

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] [4] [7].

**Lemma 4.** $\mathcal{L}(\text{DCFL}) \subseteq \mathcal{L}(\text{pmp-RP})$.

**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-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-RP-automaton $M'$ is

\begin{itemize}
  \item[\text{pumping}] we sketch the construction of $M$; the construction will also be helpful for better understanding of later results and proofs.
  \item[\text{It}] is well known that if $L'$ is a deterministic context-free language and $S$ is a symbol not in the alphabet of $L'$, then $L' \cdot \{S\}$ 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 an 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 (possibly) 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 $w = xu_1v_2y$ then $w(0) = xv_1y$). Analogously, replacing $T_2$ with $T_1$, we get the derivation tree $T_{w(2)}$ for a longer word $w(2)$ (where $w(2) = xu_1^{-1}v_2^{-1}y$). 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 $w(i+1) = xu_1^{-1}v_2^{-1}y$.

The key to a construction of the det-mon-R-automaton $M$ in [3] is the possibility to identify the leftmost subword $u_1v_2$ 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 \cite{3}, a similar proof can be based on the constructions from paper \cite{7} based on deterministic list automata.

The following corollary is a consequence of the previous theorem and Proposition \ref{prop3}.

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

**Corollary 2.**

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

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 $M_3 = (Q, \Sigma, \epsilon, S, q_0, k, \delta, Q_A, Q_R)$ be the RP-automaton with the set of states $Q = \{q_0, q_1, q_2, q_3, q_A\}$, 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

$\delta(q_0, \epsilon S) = \{(q_A, \text{HALT})\}$, $\delta(q_0, bb) = \{(q_0, \text{MVR})\}$, $\delta(q_0, ab) = \{(q_1, \text{PREPARE})\}$, $\delta(q_0, eb) = \{(q_0, \text{MVR})\}$, $\delta(q_1, ab) = \{\text{RESTART}(\lambda)\}$, $\delta(q_0, ba) = \{(q_0, \text{MVR})\}$, $\delta(q_0, bS) = \{(q_0, \text{PREPARE})\}$, $\delta(q_1, a) = \{\text{RESTART}(\lambda)\}$, $\delta(q_0, b) = \{\text{RESTART}(\lambda)\}$.

The automaton differs from the automaton $M_1$ of Example \ref{example2} only slightly. It has two new restarting states $q_1$ and $q_2$ that enable to delete any symbol to the left from the right sentinel $\$. Nevertheless, the first restarting instruction $t = (q_1, ab) \rightarrow_\delta \text{RESTART}(\lambda)$ is still two-side pumping instruction (see Example \ref{example2}). 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 $M = (Q, \Sigma, \epsilon, S, q_0, k, \delta, Q_A, Q_R)$ be a pmp-RP-automaton accepting the language $L = L(M)$. Let $t = (q_i, vu_1v_2) \rightarrow_\delta \text{RESTART}(\nu)$ be a two-side pumping restarting instruction of $M$ with pumping words (strings) $u_1, u_2$ and $vu_1v_2y \Rightarrow vu_1vy$, for some $x, y \in \Sigma^*$.

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

1. $xu_1^pvy \in L$ and $xu_1^pvy^2 \notin L$, for all $j \geq 1$, $m \geq 0$,
2. $xu_1^pv^iuy \in L$, and $xu_1^pv^juy \notin L$, for all $j \geq 1$, $m \geq 0$,
3. $xu_1^pvy \notin L$, and $xu_1^pvy^2 \in L$, for all $j \geq 1$, $m \geq 0$.
4. $xu_1^pvy \notin L$, and $xu_1^{p^j}vy \in L$, for all $j \geq 1$, $m \geq 0$.

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

We say that $M$ is a distinguishing RP-automaton if there is a distinguishing instruction $t$ for $M$. We write det-RP-automaton to denote a distinguishing pmp-RP-automaton.

**Example 4.** Let us consider the automaton $M_1 = (Q, \Sigma, \epsilon, S, q_0, k, \delta, Q_A, Q_R)$ from Example \ref{example2}. We will show that $M_1$ is a det-RP-automaton. The automaton has only one restarting instruction $t = (q_1, ab) \rightarrow_\delta \text{RESTART}(\lambda)$.

This instruction is a two-side pumping instruction with pumping strings $u_1 = a$ and $u_2 = b$ and $ab = xu_1vy \rightarrow_1 xvy = \lambda$, for $x = \lambda$, $\nu = \lambda$, $y = \lambda$. The instruction $t$ is a $(p, x, y)$-distinguishing instruction for $M_1$ for $p = 1$.

Namely, for all $j \geq 1$, $m \geq 0$ it holds

$xu_1^jvy \in L(M_1)$ and $xu_1^{p^j}vy \notin L(M_1)$,

and also

$xu_1^jvy \in L(M_1)$ and $xu_1^{p^j}vy \notin L(M_1)$.

**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 $M = (Q, \Sigma, \epsilon, S, q_0, k, \delta, Q_A, Q_R)$ is a dist-RP-automaton accepting a regular language $L = L(M)$. Since $M$ is a det-RP-automaton, it has a $(p, x, y)$-distinguishing instruction $t = (q_1, vu_1v_2) \rightarrow_\delta \text{RESTART}(\nu)$, for some $x, y, v \in \Sigma^*$, $u_1, u_2 \in \Sigma^*$, $q_i \in Q$ 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 $t$:

Case (I): From the definition, for all $m \geq 0$, $j \geq 1$ it holds

$xu_1^jvy \in L$ and $xu_1^jvy^2 \notin L$. Let $q_m$ denote the
state of the automaton A in which it reads the first symbol of the j-th copy of u1 to the right from v. That is, A is in the state qm_j after reading xu^m_jvu^p_ju^s_jy, for j ≥ 1. For m > n_A, there exist integers r, 1 ≤ r < s ≤ n_A + 1 such that qm_r = qm_s. Then, for all i ≥ 0, it holds qm_i = qm_{m+i(r-s)} and the automaton A accepts all words of the form xu^{m+i(r-s)}_ju^p_iu^s_iy. For i = p, we obtain that xu^{m+p(r-s)}_ju^p_iu^s_iy ∈ L which contradicts the assumption xu^{m+p(r-s)}_ju^p_iu^s_iy ∈ L for j = (s-r).

Case (II): From the definition, for all m ≥ 0, j ≥ 1 it holds xu^m_jvu^p_ju^s_jy ∈ L, and xu^{p-1}ju^m_jvu^p_ju^s_jy /∈ L. Let qm_j denote the state of the automaton A in which it reads the first symbol of the j-th copy of u1 to the right from x. That is, A is in the state qm_j after reading xu^{j-1}_j, for j ≥ 1. For m > n_A, there exist integers r, s, 1 ≤ r < s ≤ n_A + 1 such that qm_r = qm_s. Then, for all i ≥ 0, it holds qm_i = qm_{m+i(r-s)} and the automaton A accepts all words of the form xu^{m+i(r-s)}_ju^p_iu^s_iy. For i = p, we obtain that xu^{m+p(r-s)}_ju^p_iu^s_iy ∈ L which contradicts the assumption xu^{p-1}ju^m_jvu^p_ju^s_jy /∈ L for j = (s-r).

Analysis of conditions (III) and (IV) from Definition 1 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 (⊂ denotes the proper subset relation).

**Corollary 3.** \( L'(dist-RP) = L'(scf-dist-RP) ⊂ DCFL. \)

**Remark.** We conjecture that \( L'(scf-dist-RP) \) 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 pm-sc-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 \( \text{[9]} \) 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**