RAIRO-Inf. Theor. Appl. **39** (2005) 325-342
DOI: 10.1051/ita:2005021

HIERARCHIES OF WEAKLY MONOTONE RESTARTING AUTOMATA *

František Mráz[1] and Friedrich Otto[2]

Abstract. It is known that the weakly monotone restarting automata accept exactly the growing context-sensitive languages. We introduce a measure on the degree of weak monotonicity and show that the language classes obtained in this way form strict hierarchies for the various types of deterministic and nondeterministic restarting automata without auxiliary symbols.

Mathematics Subject Classification. 68Q10, 68Q42, 68Q45.

1. Introduction

The motivation for introducing the restarting automaton in [2] was the desire to model the so-called *analysis by reduction* of natural languages [10, 11]. This analysis consists of a stepwise simplification of a sentence in such a way that the syntactical correctness or incorrectness of the sentence is not affected. After a finite number of steps either a correct simple sentence is obtained, or an error is detected. It turned out that the restarting automaton thus obtained can handle a class of languages that is much broader than the class CFL of context-free languages.

A restarting automaton, or RRWW-automaton, M is a device with a finite state control and a read/write window of a fixed size. This window moves from left to right along a tape containing a string delimited by sentinels until M's control decides (nondeterministically) that the contents of the window should be rewritten

Keywords and phrases. Restarting automata, weak monotonicity, hierarchies.

* *The first author was partially supported by the Grant Agency of the Czech Republic, Grant-No. 201/02/1456 and Grant-No. 201/04/2102. The work of the second author was supported by a grant from the Deutsche Forschungsgemeinschaft.*

[1] Department of Computer Science, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic; mraz@ksvi.ms.mff.cuni.cz

[2] Fachbereich Mathematik/Informatik, Universität Kassel, 34109 Kassel, Germany; otto@theory.informatik.uni-kassel.de

by some shorter string. After a rewrite, M continues to move its window to the right until it either halts and accepts, or halts and rejects, which means that it has reached a configuration from which it cannot continue, or restarts, that is, it moves its window to the leftmost position, enters the initial state, and continues with the computation. Thus, each computation of M can be described by a sequence of cycles.

Also some restricted variants of the restarting automaton have been considered. There are those variants that only use the letters from the input alphabet, while in general a restarting automaton can use a finite number of auxiliary symbols in its rewrite steps. Further, a monotonicity property was introduced for RRWW-automata, which is based on the idea that from one cycle to the next in a computation the actual place where a rewriting is performed must not increase its distance from the right end of the tape. It turned out that the monotone RRWW-automata (with auxiliary symbols) characterize the class CFL, and that various restricted versions of deterministic monotone RRWW-automata (with or without auxiliary symbols) characterize the class DCFL of deterministic context-free languages [3].

In [5,6] the class CRL of Church-Rosser languages was introduced motivated by the fact that membership for these languages is decidable in linear time. In [9] it was then shown that the deterministic RRWW-automata (with auxiliary symbols) give another characterization of CRL. For the growing context-sensitive languages GCSL [1], which can be seen as the nondeterministic variants of the Church-Rosser languages [7], it was observed in [8] that they form a proper subclass of the class $\mathcal{L}$(RRWW) of languages that are accepted by RRWW-automata.

In order to obtain a characterization of GCSL in terms of restarting automata, a relaxation of the monotonicity condition for RRWW-automata was introduced in [4]. Let c be a non-negative integer. An RRWW-automaton M is called *weakly c-monotone* if in any two consecutive rewrite steps of any computation of M the places of rewriting can increase their distances from the right end of the tape by at most c positions. An RRWW-automaton M is called *weakly monotone* if there exists a non-negative integer c such that M is weakly c-monotone. The weakly monotone RRWW-automata (with auxiliary symbols) recognize exactly GCSL [4].

Here we study the influence of the degree of weak monotonicity on the expressive power of the various models of restarting automata. We focus on the restarting automata with the most transparent computations – those without auxiliary symbols. Further motivation for studying weak monotonicity comes from linguistics. The degree of weak monotonicity is a measure of across how many symbols (words of a sentence) already read before a reader must step back to the left during an analysis of a sentence in a certain (natural) language.

In the next section we restate the main definitions in short, and we prove that weak monotonicity is a decidable property of restarting automata. In Section 3 we show that by increasing the degree of weak monotonicity the expressive power of all considered variants of restarting automata without auxiliary symbols is increased. Section 4 shows that all the hierarchies obtained differ from one another, and Section 5 contains some concluding remarks.

2. Definitions

The following definition differs slightly from the one given in the literature (*cf.*, *e.g.*, [3, 9]), but the two definitions are easily seen to be equivalent to each other.

A *restarting automaton*, RRWW-automaton for short, is a one-tape machine that is described by an 8-tuple $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$, where Q is the finite set of states, Σ is the finite input alphabet, Γ is the finite tape alphabet containing Σ, $¢, \$ \notin \Gamma$ are symbols that are used as delimiters for the left and right border of the work space, respectively, $q_0 \in Q$ is the initial state, $k \geq 1$ is the size of the *read/write window*, and

$$\delta : Q \times \mathcal{PC}^{(k)} \to \mathfrak{P}\left(\left(Q \times \left(\{\mathsf{MVR}\} \cup \mathcal{PC}^{\leq(k-1)}\right)\right) \cup \{\mathsf{Restart}, \mathsf{Accept}\}\right)$$

is the *transition relation*. Here $\mathfrak{P}(S)$ denotes the powerset of the set S, and $\mathcal{PC}^{(k)}$ is the set of *possible contents* of the read/write window of M, where

$$\mathcal{PC}^{(i)} := \left(¢ \cdot \Gamma^{i-1}\right) \cup \Gamma^i \cup \left(\Gamma^{\leq i-1} \cdot \$\right) \cup \left(¢ \cdot \Gamma^{\leq i-2} \cdot \$\right) \quad (i \geq 0),$$

and

$$\Gamma^{\leq n} := \bigcup_{i=0}^{n} \Gamma^i \quad \text{and} \quad \mathcal{PC}^{\leq(k-1)} := \bigcup_{i=0}^{k-1} \mathcal{PC}^{(i)}.$$

The transition relation describes four different types of transition steps:

(1) A *move-right step* is of the form $(q', \mathsf{MVR}) \in \delta(q, u)$, where $q, q' \in Q$ and $u \in \mathcal{PC}^{(k)}$, $u \neq \$$. If M is in state q and sees the string u in its read/write window, then this move-right step causes M to shift the read/write window one position to the right and to enter state q'. However, if the contents u of the read/write window is only the right border marker \$, then no shift to the right is possible.

(2) A *rewrite step* is of the form $(q', v) \in \delta(q, u)$, where $q, q' \in Q$, $u \in \mathcal{PC}^{(k)}$, $u \neq \$$, and $v \in \mathcal{PC}^{\leq(k-1)}$ such that $|v| < |u|$. It causes M to replace the contents u of the read/write window by the string v and to enter state q'. Further, the read/write window is placed immediately to the right of the string v. However, some additional restrictions apply in that the border markers ¢ and \$ must not disappear from the tape nor that new occurrences of these markers are created. Further, the read/write window must not move across the right border marker \$, that is, if v ends in \$, then the read/write window is placed on the \$-symbol.

(3) A *restart step* is of the form $\mathsf{Restart} \in \delta(q, u)$, where $q \in Q$ and $u \in \mathcal{PC}^{(k)}$. It causes M to move its read/write window to the left end of the tape, so that the first symbol it sees is the left border marker ¢, and to reenter the initial state q_0.

(4) An *accept step* is of the form $\mathsf{Accept} \in \delta(q, u)$, where $q \in Q$ and $u \in \mathcal{PC}^{(k)}$. It causes M to halt and accept.

If $\delta(q,u) = \emptyset$ for some $q \in Q$ and $u \in \mathcal{PC}^{(k)}$, then M necessarily halts, and we say that M *rejects* in this situation.

A *configuration* of M is a string $\alpha q \beta$, where $q \in Q \cup \{\mathsf{Accept}\}$, and either $\alpha = \varepsilon$ and $\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{¢\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\$\}$; here $q \in Q$ 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 of β when $|\beta| \leq k$. A *restarting configuration* is of the form $q_0¢w\$$, where $w \in \Gamma^*$; if $w \in \Sigma^*$, then $q_0¢w\$$ is an *initial configuration*. A configuration containing the "command" Accept is called an *accepting configuration*, and a configuration of the form $\alpha q \beta$, where the read/write window contains the prefix β_1 of β and $\delta(q,\beta_1) = \emptyset$, is called a *rejecting configuration*. The accepting and the rejecting configurations together are the *halting configurations* of M.

Obviously, each computation of M proceeds in cycles. Starting from an initial configuration $q_0¢w\$$, the head moves right, while move-right and rewrite steps are executed, until finally a restart step takes M back into a restarting configuration of the form $q_0¢w_1\$$. By $q_0¢w\$ \vdash_M^c q_0¢w_1\$$ we denote the execution of a complete cycle, and $\vdash_M^{c^*}$ denotes the reflexive and transitive closure of this relation. It is required that in each cycle *exactly one* rewrite step is executed. As by a rewrite step the contents of the tape is shortened, only a linear number of cycles can be executed within any computation. That part of a computation of M that follows after the execution of the last restart is called the *tail* of the computation. It contains at most a single application of a rewrite step.

An input $w \in \Sigma^*$ is accepted by M, if there exists a computation of M which starts with the initial configuration $q_0¢w\$$, and which finally ends with executing an accept step. By $L(M)$ we denote the *language accepted by* M, and $\mathcal{L}(\mathsf{RRWW})$ will denote the class of languages that are accepted by RRWW-automata.

It is easily seen that each RRWW-automaton can be modified such that it makes an accept or a restart step only when it sees the right border marker \$ in its read/write window. This means that in each cycle of each computation and also during the tail of each accepting computation the read/write window moves all the way to the right before a restart is made, respectively, before the machine halts and accepts.

Based on this fact each cycle (and also the tail) of a computation of an RRWW-automaton M consists of three phases. Accordingly, the transition relation of an RRWW-automaton can be described through a finite sequence of so-called *meta-instructions* [8] of the form $(R_1, u \to v, R_2)$, where R_1 and R_2 are regular languages, called the *regular constraints* of this instruction, and u and v are strings such that $|u| > |v|$, where $u \to v$ stands for a rewrite step. On trying to execute this meta-instruction M will get stuck (and so reject) starting from the configuration $q_0¢w\$$, if w does not admit a factorization of the form $w = w_1uw_2$ such that $¢w_1 \in R_1$ and $w_2\$ \in R_2$. On the other hand, if w does have a factorization of this form, then one such factorization is chosen nondeterministically, and $q_0¢w\$$ is transformed into $q_0¢w_1vw_2\$$. In order to be able to also describe the tails of accepting computations we use meta-instructions containing accept instructions

of the form $(¢ \cdot R \cdot \$, \mathsf{Accept})$, where the strings from the regular language R are accepted by M in tail computations.

We illustrate the above definition by an example.

Example 2.1. Let $L_{\exp} := \{\, a^{2^n} \mid n \geq 0 \,\}$. We present an RRWW-automaton $M_{\exp}$ that accepts this language. It has input alphabet $\{a\}$, tape alphabet $\{a, A\}$, and a read/write window of size 3, and it is described through the following sequence of meta-instructions:

(1) $(¢ \cdot a^*,\ aa\$ \to A\$,\ \varepsilon)$;
(2) $(¢ \cdot a^*,\ aaA \to AA,\ A^* \cdot \$)$;
(3) $(¢ \cdot A^*,\ AA\$ \to a\$,\ \varepsilon)$;
(4) $(¢ \cdot A^*,\ AAa \to aa,\ a^* \cdot \$)$.
(5) $(¢ \cdot a \cdot \$,\ \mathsf{Accept})$;
(6) $(¢ \cdot A \cdot \$,\ \mathsf{Accept})$.

Given an input of the form $w := a^{2^n}$ $(n \geq 2)$, $M_{\exp}$ will execute the following sequence of cycles:

$$q_0¢a^{2^n}\$ \vdash^{c^*}_{M_{\exp}} q_0¢A^{2^{n-1}}\$ \vdash^{c^*}_{M_{\exp}} q_0¢a^{2^{n-2}}\$.$$

This computation continues until the restarting configuration $q_0¢a\$$ or $q_0¢A\$$ is reached, which then leads to acceptance. On the other hand, for an input of the form $w := a^m$, where $m = 2^{n_0} \cdot (2n_1 + 1)$ for some integers $n_0 \geq 0$ and $n_1 \geq 1$, $M_{\exp}$ will reach the restarting configuration $q_0¢a^{2n_1+1}\$$ or $q_0¢A^{2n_1+1}\$$ after a number of cycles, and then it will get stuck and therewith reject in the configuration $q_0¢aA^{n_1}\$$ or $q_0¢Aa^{n_1}\$$, respectively. This shows that indeed $L(M_{\exp}) = L_{\exp}$ holds.

A restarting automaton is called an RWW-*automaton* if it makes a restart immediately after performing a rewrite operation. Hence, a cycle of a computation of an RWW-automaton consists of two phases only. Accordingly, the transition relation of an RWW-automaton can be described by a finite sequence of *meta-instructions* of the forms $(R, u \to v)$ and $(¢ \cdot R \cdot \$, \mathsf{Accept})$.

An R(R)WW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ is *deterministic* if its transition relation is a (partial) function

$$\delta : Q \times \mathcal{PC}^{(k)} \to \Big(Q \times \Big(\{\mathsf{MVR}\} \cup \mathcal{PC}^{\leq (k-1)}\Big)\Big) \cup \{\mathsf{Restart}, \mathsf{Accept}\}.$$

It is called an R(R)W-*automaton* if it is an R(R)WW-automaton for which the tape alphabet Γ coincides with the input alphabet Σ, that is, if no auxiliary symbols are available. Finally, it is an R(R)-*automaton* if it is an R(R)W-automaton for which the right-hand side v of each rewrite step $(q', v) \in \delta(q, u)$ is a scattered substring of the left-hand side u.

We will often (implicitly) use the following properties [3].

Lemma 2.2 (The error preserving property)**.** *Let $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ be an* RRWW*-automaton, and let u, v be strings from Σ^*. If $q_0¢u\$ \vdash^{c^*}_M q_0¢v\$$ and $u \notin L(M)$, then $v \notin L(M)$.*

Lemma 2.3 (The correctness preserving property). *Let $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ be an* RRWW*-automaton, and let u, v be strings from Σ^*. If $q_0¢u\$ \vdash_M^{c^*} q_0¢v\$$ is an initial segment of an accepting computation of M, then $v \in L(M)$.*

If M is a deterministic RRWW-automaton with input alphabet Σ, then the correctness preserving property implies that, for strings $u, v \in \Sigma^*$, if $u \in L(M)$ and $q_0¢u\$ \vdash_M^{c^*} q_0¢v\$$, then also $v \in L(M)$.

Obviously the RRWW-automaton $M_{\rm exp}$ above is deterministic. On the other hand, based on Lemma 2.3 it is easily seen that the language $L_{\rm exp}$ cannot be accepted by any RRW-automaton.

In [3] a notion of monotonicity is considered for restarting automata. Let M be an RRWW-automaton. Each computation of M can be described by a sequence of cycles $C_1, C_2, \ldots, C_n$, where C_1 starts with an initial configuration of M, and C_n is the last cycle, which is followed by the tail of the computation. Each cycle C_i contains a unique configuration of the form $¢xquy\$$ such that q is a state and $u \to v$ is the rewrite step applied during this cycle. By $D_r(C_i)$ we denote the *r-distance* $|y\$|$ of this cycle. The sequence of cycles $C_1, C_2, \ldots, C_n$ is called *monotone* if $D_r(C_1) \geq D_r(C_2) \geq \ldots \geq D_r(C_n)$ holds, and the RRWW-automaton M is called *monotone* if all its computations are monotone.

In [3] it is shown that all variants of deterministic monotone restarting automata accept exactly the deterministic context-free languages. For the nondeterministic restarting automata it turned out that the use of auxiliary symbols is necessary to obtain a characterization of the class of context-free languages.

In order to derive a characterization of the class GCSL of growing context-sensitive languages in terms of restarting automata, the notion of weak monotonicity was introduced in [4]. Let M be an RRWW-automaton, and let $c \geq 0$ be an integer. We say that a sequence of cycles $C_1, C_2, \ldots, C_n$ of M is *weakly c-monotone*, if $D_r(C_{i+1}) \leq D_r(C_i) + c$ holds for all $i = 1, 2, \ldots, n-1$. A computation of M is called *weakly c-monotone* if the corresponding sequence of cycles is weakly c-monotone. Observe that the tail of the computation is not taken into account. Further, the RRWW-automaton M is called *weakly c-monotone* if, for each restarting configuration $q_0¢w\$$ of M, each computation of M starting with $q_0¢w\$$ is weakly c-monotone. Note that here we do not only consider computations that start with an initial configuration, but that we explicitly consider all computations that start with a restarting configuration. Finally, we say that M is *weakly monotone*, if there exists a constant $c \geq 0$ such that M is weakly c-monotone. The prefixes w(c)mon- and wmon- are used to denote the corresponding classes of restarting automata.

It is easily seen that the above RRWW-automaton $M_{\rm exp}$ is weakly 1-monotone. Actually, for deterministic RRWW-automata the following general observation holds.

Lemma 2.4. *Each deterministic* RRWW*-automaton with window size k is weakly c-monotone for some $c \leq k - 2$.*

Proof. Assume that the rewriting is done on the tape with contents $¢xuy\$$, where u is the factor of size k that is being rewritten. Hence, the new tape contents is $¢xvy\$$ for some v of length $|v| \leq k-1$. As the RRWW-automaton considered is deterministic, the next rewrite operation cannot occur before the automaton sees at least the first letter of $vy\$$, that is, the distance from the right end of the tape is increased by at most $|v| - 1 \leq k-2$. Hence, the deterministic restarting automaton considered is weakly c-monotone for some $c \leq k-2$. □

Thus, for deterministic R(R)WW-automata the above weak monotonicity condition is always satisfied, that is, the characterization of the class CRL of Church-Rosser languages presented in [9] can be stated as

$$\mathsf{CRL} = \mathcal{L}(\text{det-wmon-RWW}) = \mathcal{L}(\text{det-wmon-RRWW}).$$

Hence, it is only for the various nondeterministic restarting automata that these additional restrictions make a difference. In [4] it is shown that

$$\mathsf{GCSL} = \mathcal{L}(\text{wmon-RWW}) = \mathcal{L}(\text{wmon-RRWW}) \subset \mathcal{L}(\text{RWW}),$$

where the inclusion is known to be proper.

In [3] it is shown that it is decidable whether an RRWW-automaton is monotone. Actually we can generalize this result to weak monotonicity as follows. Lemma 2.4 shows that, for a deterministic restarting automaton, the degree of weak monotonicity is restricted by the size of the read/write window. For nondeterministic restarting automata we also obtain an upper bound for the degree of weak monotonicity, albeit a much larger one.

Lemma 2.5. *Let $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ be a weakly monotone* RRWW*-automaton. Then M is weakly c-monotone for some constant $c < |Q|^2 \cdot |\Gamma|^k + 2k$.*

Proof. Let C_1, C_2 be two successive cycles of a computation of M such that $D_r(C_2) = D_r(C_1) + c$, that is, C_1 contains a rewrite step $¢xq_1uy\$ \vdash_M ¢xvq_1'y\$$, and C_2 contains a rewrite step $¢x'q_2u'y'\$ \vdash_M ¢x'v'q_2'y'\$$, where $q_1, q_1', q_2, q_2' \in Q$, $xvy = x'u'y'$, and $|y\$| + c = |y'\$|$.

Assume that $c \geq |Q|^2 \cdot |\Gamma|^k + 2k$. Then $x = x_1x_2$, where $x'u' = x_1$, $x_2vy = y'$, and $|x_2v| = c$ implying that $|x_2| > |Q|^2 \cdot |\Gamma|^k + k$. As we can assume that M scans its tape completely from left to right during each cycle, we see that during both cycles C_1 and C_2, the automaton M executes $|x_2|$ MVR-steps while moving its read/write window across x_2. Let $x_2 = a_1a_2 \dots a_n$, where $a_1, \dots, a_n \in \Gamma$ and $n := |x_2|$. With each symbol a_i of x_2, we can associate a pair of states (p_1^i, p_2^i) by choosing p_j^i as the state of M at the moment when a_i is the first symbol in the read/write window during the cycle C_j $(j = 1, 2)$. As $|x_2| > |Q|^2 \cdot |\Gamma|^k + k$, we see that x_2 admits a factorization of the form $x_2 = z_1az_2az_3$, where $a \in \Gamma$, $z_1, z_2, z_3 \in \Gamma^*$, and $|z_3| \geq k-1$, such that both distinguished occurrences of a have the same associated pair of states (p, p'), and the read/write window has the same contents when these occurrences of the letter a are in the first position of the read/write window. Hence, for each $m \geq 0$, the following is a valid

computation of M:

$$\begin{array}{llll} q_0\¢x_1z_1(az_2)^maz_3uy\$ & \vdash_M^* & \¢x_1z_1p(az_2)^maz_3uy\$ & \vdash_M^* \\ \¢x_1z_1(az_2)^mpaz_3uy\$ & \vdash_M^* & q_0\¢x_1z_1(az_2)^maz_3vy\$ & = \\ q_0\¢x'u'z_1(az_2)^maz_3vy\$ & \vdash_M^* & \¢x'v'z_1p'(az_2)^maz_3vy\$ & \vdash_M^* \\ \¢x'v'z_1(az_2)^mp'az_3vy\$ & \vdash_M^* & q_0\¢x'v'z_1(az_2)^maz_3vy\$, & \end{array}$$

where the first rewrite step has r-distance $|y|+1$, while the second rewrite step has r-distance $|z_1|+(1+|z_2|)\cdot m+|z_3vy|+2$. As this holds for all $m \geq 0$, we conclude that M is not weakly j-monotone for any integer j, that is, M is not weakly monotone. This contradiction proves the upper bound for the degree of weak monotonicity of M. □

Further, we have the following decidability result.

Theorem 2.6. *The following problem can be solved algorithmically:*

INSTANCE : *An* RRWW*-automaton M, and a constant $c \in \mathbb{N}$.*
QUESTION : *Is M weakly c-monotone?*

Proof. Given an RRWW-automaton M and a constant $c \in \mathbb{N}$, a nondeterministic finite-state acceptor $A_{M,c}$ can be constructed such that the language $L(A_{M,c})$ accepted by $A_{M,c}$ is non-empty if and only if M admits a computation $C_1 \vdash_M^c C_2$ satisfying $D_r(C_2) > D_r(C_1)+c$, that is, if and only if M is not weakly c-monotone. This construction is a slight variant of the corresponding construction used in [3] to show that monotonicity of RRWW-automata is decidable. □

Together with the above lemma bounding the degree of weak monotonicity, this theorem has the following consequence.

Corollary 2.7. *Given a restarting automaton M, it is decidable whether M is weakly monotone. In the affirmative the smallest integer c can be determined for which M is weakly c-monotone.*

3. Hierarchies with respect to the degree of weak monotonicity

In this section we present a sequence of sample languages

$$L_d^{(i)} \in \mathcal{L}(\mathsf{det\text{-}w}(i)\mathsf{mon\text{-}R}) \setminus \mathcal{L}(\mathsf{w}(i-1)\mathsf{mon\text{-}RRW})\ (i \geq 1).$$

This shows that by increasing the degree of weak monotonicity we increase the expressive power of all deterministic and nondeterministic variants of restarting

automata without auxiliary symbols. We construct these languages in two steps. First we give a sequence of languages

$$L_c^{(i)} \in \mathcal{L}(\mathsf{det\text{-}w}(i)\mathsf{mon\text{-}R}) \setminus \mathcal{L}(\mathsf{det\text{-}w}(i-1)\mathsf{mon\text{-}RRW}) \ (i \geq 1).$$

Then we modify these languages to get the languages $L_d^{(i)}$.

For each integer $i \geq 1$, we define the language $L_c^{(i)} \subset \{a, b, c, d, e\}^*$ as

$$L_c^{(i)} := L_{c,1}^{(i)} \cup L_{c,2}^{(i)},$$

where

$$\begin{aligned} L_{c,1}^{(i)} &:= \left\{ a^n \left(b^i c^i\right)^{n-k} \left(b^i\right)^k d \mid n \geq k \geq 0 \right\}, \text{ and} \\ L_{c,2}^{(i)} &:= \left\{ a^n \left(b^i c^i\right)^{m-k} \left(c^i\right)^k e \mid n \geq 0,\ m > 2n \text{ and } m \geq k \geq 0 \right\}. \end{aligned}$$

Notice that, for each $w \in L_c^{(i)}$, if w contains a substring b^{i+1}, then $w \in L_{c,1}^{(i)}$. Similarly, if w contains a substring c^{i+1} or ac, then $w \in L_{c,2}^{(i)}$. Obviously, $L_c^{(i)}$ is a context-free language for each $i \geq 1$.

The idea behind the languages $L_c^{(i)}$ is as follows: the strings of each of these languages can be cut in the middle by two types of operations depending on whether w belongs to $L_{c,1}^{(i)}$ or to $L_{c,2}^{(i)}$. If w starts with $a^n(b^i c^i)^m$ for some $m > 0$, and if it contains b^{i+1} as a substring or ends in d, or if it contains c^{i+1} as a substring or ends in e, then this information, which tells the type of w, can be passed from the right end of w to the rightmost a by a deterministic $\mathsf{w}(i)\mathsf{mon}$-computation.

Lemma 3.1. *For each $i \geq 1$, $L_c^{(i)} \in \mathcal{L}(\mathsf{det\text{-}w}(i)\mathsf{mon\text{-}R})$.*

Proof. Let i be a positive integer. We present an R-automaton $M_c^{(i)}$ with a read/write window of size $3i + 2$ through a sequence of meta-instructions:

(1) $\left(\text{¢}{\cdot}a^+{\cdot}\left(b^i c^i\right)^*, b^i c^i{\cdot}d{\cdot}\$ \to b^i{\cdot}d{\cdot}\$\right)$;
(2) $\left(\text{¢}{\cdot}a^*{\cdot}\left(b^i c^i\right)^*, b^i c^i{\cdot}e{\cdot}\$ \to c^i{\cdot}e{\cdot}\$\right)$;
(3) $\left(\text{¢}{\cdot}a^+{\cdot}\left(b^i c^i\right)^*, b^i c^i{\cdot}b^i{\cdot}d \to b^i{\cdot}b^i{\cdot}d\right)$;
(4) $\left(\text{¢}{\cdot}a^+{\cdot}\left(b^i c^i\right)^*, b^i c^i{\cdot}b^i{\cdot}b \to b^i{\cdot}b^i{\cdot}b\right)$;
(5) $\left(\text{¢}{\cdot}a^+, a{\cdot}b^i{\cdot}b^i{\cdot}b \to b^i{\cdot}b\right)$;
(6) $\left(\text{¢}{\cdot}a^+, a{\cdot}b^i{\cdot}b^i{\cdot}d{\cdot}\$ \to b^i{\cdot}d{\cdot}\$\right)$;
(7) $\left(\text{¢}, a{\cdot}b^i{\cdot}d{\cdot}\$ \to d{\cdot}\$\right)$;
(8) $\left(\text{¢}{\cdot}d{\cdot}\$, \mathsf{Accept}\right)$;
(9) $\left(\text{¢}{\cdot}a^*{\cdot}\left(b^i c^i\right)^*, b^i c^i{\cdot}c^i{\cdot}e \to c^i{\cdot}c^i{\cdot}e\right)$;
(10) $\left(\text{¢}{\cdot}a^*{\cdot}\left(b^i c^i\right)^*, b^i c^i{\cdot}c^i{\cdot}c \to c^i{\cdot}c^i{\cdot}c\right)$;
(11) $\left(\text{¢}{\cdot}a^*, a{\cdot}c^i{\cdot}c^i{\cdot}c \to c\right)$;
(12) $\left(\text{¢}{\cdot}\left(c^i\right)^+{\cdot}e{\cdot}\$, \mathsf{Accept}\right)$.

To any tape contents from $\{a, b, c, d, e\}^*$, at most one meta-instruction is applicable. Hence, $M_c^{(i)}$ is a deterministic R-automaton. It is easily seen that each

string w accepted by $M_c^{(i)}$ is either of the form $a^* \cdot (b^i c^i)^* \cdot (b^i)^* \cdot d$ or of the form $a^* \cdot (b^i c^i)^* \cdot (c^i)^* \cdot e$. Just observe the following:

- The only strings that are accepted by $M_c^{(i)}$ in a tail computation, that is, without making a restart, are d (8) and $(c^i)^k e$, $k \geq 1$ (12).
- The symbols b and c are deleted in blocks of length i only.
- Whenever $M_c^{(i)}$ executes a rewrite step, then the maximal proper prefix of the tape contents read is of the form $a^+ \cdot (b^i c^i)^* \cdot (b^i)^*$ (3 – 5) or of the form $a^* \cdot (b^i c^i)^* \cdot (c^i)^*$ (9 – 11), or the tape contains a string of the form $a^+ \cdot (b^i c^i)^+ \cdot d$ (1), of the form $a^* \cdot (b^i c^i)^+ \cdot e$ (2), of the form $a^+ \cdot b^i \cdot b^i \cdot d$ (6) or of the form $a \cdot b^i \cdot d$ (7).
- The form of the meta-instructions guarantees that there are not any c's to the right of a factor $b^i \cdot b$ in w, and that there are no b's to the right of a factor ac or $c^i \cdot c$ in w.

Now the following two results hold.

Claim 1. A string $w \in a^* \cdot (b^i c^i)^* \cdot (b^i)^* \cdot d$ is accepted by $M_c^{(i)}$ iff $w \in L_{c,1}^{(i)}$, and the computation of $M_c^{(i)}$ on w is weakly i-monotone.

Proof. Let $w = a^n (b^i c^i)^m (b^i)^p d$, where $m, n, p \geq 0$. We distinguish between three cases:

$n = 0$: Then the only meta-instruction that may be applicable is (8), and $M_c^{(i)}$ accepts w iff $w = d \in L_{c,1}^{(i)}$.

$n > 0$ and $m = 0$: if $p = 0$, then $M_c^{(i)}$ has no applicable instruction, and hence, w is not accepted. In case $p > 1$, $M_c^{(i)}$ repeatedly deletes ab^i using meta-instructions (5–7) until it either obtains d and accepts or until it cannot proceed and rejects. Thus, w is accepted iff $n = p$, that is, $w \in L_{c,1}^{(i)}$. Moreover, the corresponding part of the computation on w is monotone.

$n > 0$ and $m > 0$: if $p = 0$, then $M_c^{(i)}$ applies instruction (1). If $p = 1$, then $M_c^{(i)}$ uses meta-instruction (3). For $p > 1$, $M_c^{(i)}$ repeatedly uses meta-instruction (4) until it rejects or until it obtains a string of the form $a^n (b^i)^{p+m} d$, which is one of the above cases. Each application of meta-instruction (4) causes an increase of the r-distance by i. Thus, the corresponding part of the computation is weakly i-monotone. □

Claim 2. A string $w \in a^* \cdot (b^i c^i)^* \cdot (c^i)^* \cdot e$ is accepted by $M_c^{(i)}$ iff $w \in L_{c,2}^{(i)}$, and the computation of $M_c^{(i)}$ on w is weakly i-monotone.

Proof. We omit the detailed proof of this claim, as it closely follows the proof of Claim 1. The differences are in

- deleting blocks of the form c^i instead of b^i for shifting the information that w is a string from $L_{c,2}^{(i)}$ to the left;
- cutting "in the middle" deletes $ac^i c^i$ (meta-instruction (11));
- the strings directly accepted by $M_c^{(i)}$ are of the form $(c^i)^+ \cdot e$ (12). □

If the input w is neither of the form described in Claim 1 nor of the form described in Claim 2, then $M_c^{(i)}$ will not accept, but it is easily seen that the corresponding computation is still weakly i-monotone. It follows that $M_c^{(i)}$ is a det-w(i)mon-R-automaton for the language $L_c^{(i)}$. □

Next we will show that the degree i of weak monotonicity is necessary also for a det-RRW-automaton accepting $L_c^{(i)}$.

Lemma 3.2. *For each $i \geq 1$, $L_c^{(i)} \notin \mathcal{L}(\text{det-w}(i-1)\text{mon-RRW})$.*

Proof. Let i be a positive integer. We will show that $L_c^{(i)}$ is not accepted by any det-w($i-1$)mon-RRW-automaton. Suppose that there were such an automaton M'. By Lemma 2.3 each cycle of an accepting computation of M' transforms a string that belongs to the language $L_c^{(i)}$ into another, shorter string that also belongs to this language. Hence, it follows from the definition of $L_c^{(i)}$ that M' cannot rewrite less than i symbols in any cycle of an accepting computation, implying that the size k of the read/write window of M' is at least i.

Let p be the number of states of M', let $n := (p+k)!$, and let $w := a^n(b^ic^i)^nd$. It is clear that M' cannot accept w in a tail computation, that is, without performing any restart steps, as in tail computations only regular languages are accepted. We consider the first cycle C of an accepting computation of M' on input w. Through this cycle M' transforms w into a shorter string also belonging to $L_c^{(i)}$ (Lem. 2.3). Thus, during this cycle M' deletes m_1 symbols a, $i{\cdot}m_1$ symbols b, and $i{\cdot}m_2$ symbols c for some m_1, $m_2 \geq 0$. We distinguish between two cases.

Case 1 ($m_1 > 0$). Then w is cut at the border between the blocks a^n and $(b^ic^i)^n$ and $m_1 = m_2$ follows. After a rewrite an RRW-automaton reads the remaining part of the tape completely before it restarts. During this process it behaves like a finite-state acceptor. Thus, as the exponent n is large, there will be two occurrences of blocks b^ic^i in this part of the tape such that, while reading the first b of these particular blocks, M' will be in the same internal state. Let the distance between the first symbols of the closest such two blocks be $2i \cdot m$. Certainly, $p \geq m \geq 1$, and n is divisible by m. Thus, M' will perform the same rewriting on strings of the form $a^n(b^ic^i)^{n+jm}e$ for all integers $j \geq 0$. However, when it sees the symbol e at the right end of the string, M' may behave differently than on w. It may

(1) either reject – then M' will also reject the string $a^n(b^ic^i)^{n+2nm}e \in L_c^{(i)}$, which is a contradiction, or
(2) accept – then M' will also accept the string $a^n(b^ic^i)^{n+0\cdot m}e \notin L_c^{(i)}$, which is a contradiction, or
(3) restart – then M' will also restart on $a^n(b^ic^i)^{n+\frac{n}{m}\cdot m}e = a^n(b^ic^i)^{2n}e \notin L_c^{(i)}$, which yields $a^{n-m_1}(b^ic^i)^{2n-m_1}e \in L_c^{(i)}$, contradicting the error preserving property (Lem. 2.2).

Case 2 ($m_1 = 0$). In this case $m_2 > 0$ and the only option for M' is to delete the m_2 rightmost blocks of c^i. The r-distance of the cycle C is at most 1 (the right

sentinel need not be scanned). Let C_x (for $x = 1, 2, \ldots$) denote the x-th cycle of the computation of M' on w, and let w_x denote the resulting string after C_x, that is, $C = C_1$ and $D_r(C_1) \leq 1$. Due to the weak $(i-1)$-monotonicity of M', we have an upper bound for the r-distance

$$D_r(C_x) \leq (x-1)(i-1) + 1 = xi + 2 - (x+i). \tag{1}$$

However, in these cycles only whole blocks c^i (the rightmost ones in the current string) can be deleted. In each of these cycles, M' deletes at least i symbols. Hence, after $x-1$ cycles the resulting string w_{x-1} contains at most $n - x + 1$ blocks of c^i, and w_{x-1} has suffix $(b^i)^{x-1}d$. From this it follows that

$$D_r(C_x) \geq i + (x-1)i + 2 - k = xi + 2 - k. \tag{2}$$

These two bounds contradict for $x > k - i$. □

Thus, we have the following result.

Corollary 3.3. *For each $i \geq 1$,*

$$L_c^{(i)} \in \mathcal{L}(\mathsf{det\text{-}w}(i)\mathsf{mon\text{-}R}) \setminus \mathcal{L}(\mathsf{det\text{-}w}(i-1)\mathsf{mon\text{-}RRW}).$$

While $L_c^{(i)}$ is not accepted by any deterministic weakly $(i-1)$-monotone RRW-automaton, it is accepted by a nondeterministic RR-automaton that is monotone.

Lemma 3.4. *For each $i \geq 1$, $L_c^{(i)} \in \mathcal{L}(\mathsf{mon\text{-}RR})$.*

Proof. Let $i \geq 1$, and let $\overline{M}_c^{(i)}$ be the RR-automaton that works as follows:

Case 1. On strings that do not have a prefix of the form $a^j b^i c^i$ $(j > 0)$, $\overline{M}_c^{(i)}$ works in the same way as $M_c^{(i)}$, which gives a monotone computation.

Case 2. On a string w with prefix $a^j b^i c^i$ $(j > 0)$, $\overline{M}_c^{(i)}$ nondeterministically guesses that w is from $L_{c,1}^{(i)}$ (that is, ends with d) or that w is from $L_{c,2}^{(i)}$ (that is, ends with e). In the first case $\overline{M}_c^{(i)}$ deletes $ab^i c^i$ and verifies that the rest of the string is of the form $(b^i c^i)^* \cdot (b^i)^* \cdot d$. In the affirmative it restarts, otherwise it rejects; in the second case, $\overline{M}_c^{(i)}$ deletes $ab^i c^i b^i c^i$ or $ab^i c^i c^i$ (if neither is possible, then $\overline{M}_c^{(i)}$ halts immediately without accepting) and verifies that the rest of the string is of the form $(b^i c^i)^* \cdot (c^i)^* \cdot e$ or $(c^i)^* \cdot e$, respectively. In the affirmative it restarts, otherwise it rejects.

In both cases the rewriting is performed on the border between the a-syllable and the $(b^i c^i)$-syllable, and so also these computations of $\overline{M}_c^{(i)}$ are monotone.

Thus, $\overline{M}_c^{(i)}$ is a monotone RR-automaton that accepts the language $L_c^{(i)}$. □

This lemma shows that we need to modify the languages $L_c^{(i)}$ in order to obtain the intended example languages $L_d^{(i)}$. For each $i \geq 1$, we define four auxiliary

languages as follows:

$$L_{a1}^{(i)} := \left\{ a^n \left(b^i c^i\right)^n \mid n \geq 0 \right\}, \qquad L_{de1} := \left\{ d^r e^s \mid r > s \geq 0, r+s > 1 \right\},$$
$$L_{a2}^{(i)} := \left\{ a^n \left(b^i c^i\right)^m \mid m > 2n \geq 0 \right\}, \quad L_{de2} := \left\{ d^r e^s \mid 0 \leq r < s, r+s > 1 \right\}.$$

Now the language $L_d^{(i)}$ is defined as

$$L_d^{(i)} := L_c^{(i)} \cup L_{a1}^{(i)} \cdot L_{de1} \cup L_{a2}^{(i)} \cdot L_{de2}.$$

Clearly, for each $i \geq 1$, the language $L_d^{(i)}$ is context-free. Notice further that for any $w = uv \in L_d^{(i)}$, where $u \in \{a,b,c\}^*$ and $v \in \{d,e\}^*$, an occurrence of the substring b^{i+1} in u implies $w \in L_{c,1}^{(i)}$. Similarly, an occurrence of the substring c^{i+1} or ac implies $w \in L_{c,2}^{(i)}$. Moreover, $L_{a1}^{(i)} \cdot \{d\} \subset L_{c,1}^{(i)}$ and $L_{a2}^{(i)} \cdot \{e\} \subset L_{c,2}^{(i)}$.

Lemma 3.5. *For each $i \geq 1$, $L_d^{(i)} \in \mathcal{L}(\mathsf{det\text{-}w}(i)\mathsf{mon\text{-}R})$.*

Proof. By adding the meta-instructions

(d1) $(\mathfrak{c} \cdot a^* \cdot (b^i c^i)^* \cdot d^*, dee \to e)$; (d3) $(\mathfrak{c} \cdot a^* \cdot (b^i c^i)^* \cdot d^*, dd\$ \to d\$)$;

(d2) $(\mathfrak{c} \cdot a^* \cdot (b^i c^i)^* \cdot d^*, dde\$ \to d\$)$; (d4) $(\mathfrak{c} \cdot a^* \cdot (b^i c^i)^*, ee \to e)$;

to those of $M_c^{(i)}$, an R-automaton $M_d^{(i)}$ is obtained. These additional instructions enable $M_d^{(i)}$ to reduce any string from $L_d^{(i)} \setminus L_c^{(i)}$ to a member of $a^* \cdot (b^i c^i)^* \cdot (d+e)$. The meta-instructions of $M_d^{(i)}$ are pairwise incompatible, which shows that $M_d^{(i)}$ is a deterministic R-automaton.

It is easily seen that $M_d^{(i)}$ accepts exactly the strings accepted by $M_c^{(i)}$ and those strings that can be reduced to a string from $L(M_c^{(i)})$ by the instructions (d1)–(d4). These additional instructions can be applied only to strings of the form uv, where $u \in a^* \cdot (b^i c^i)^*$, $v \in \{d,e\}^*$, and $|v| > 1$. Further, $v \in L_{de1}$ iff $q_0 \mathfrak{c} uv\$ \vdash_{M_d^{(i)}}^{c^*} q_0 \mathfrak{c} ud\$$, and $v \in L_{de2}$ iff $q_0 \mathfrak{c} uv\$ \vdash_{M_d^{(i)}}^{c^*} q_0 \mathfrak{c} ue\$$.

Finally, $a^* \cdot (b^i c^i)^* \cdot d \cap L_c^{(i)} = L_{a1} \cdot \{d\}$ and $a^* \cdot (b^i c^i)^* \cdot e \cap L_c^{(i)} = L_{a2} \cdot \{e\}$. It follows that $M_d^{(i)}$ is a $\mathsf{det\text{-}w}(i)\mathsf{mon\text{-}R}$-automaton for $L_d^{(i)}$. □

In fact, we have the following result.

Theorem 3.6. *For each $i \geq 1$, $L_d^{(i)} \in \mathcal{L}(\mathsf{det\text{-}w}(i)\mathsf{mon\text{-}R}) \setminus \mathcal{L}(\mathsf{w}(i-1)\mathsf{mon\text{-}RRW})$.*

Proof. It remains to show that $L_d^{(i)}$ is not accepted by any $\mathsf{w}(i-1)\mathsf{mon\text{-}RRW}$-automaton. So assume that there is a $\mathsf{w}(i-1)\mathsf{mon\text{-}RRW}$-automaton M' for $L_d^{(i)}$ for some $i \geq 1$. The size k of the read/write window of M' is at least i, because in a cycle of an accepting computation on an input from $L_c^{(i)}$, M' cannot rewrite less than i symbols.

Let p be the number of states of M', let $n := (p+k)!$, and let

$$w := a^n \left(b^i c^i\right)^n d^{n+1} e^n \in L_{a1} \cdot L_{de1}.$$

We consider the first cycle C of an accepting computation of M' on w.

In C, M' cannot rewrite on the border of a^n and $(b^i c^i)^n$. Otherwise, it must delete $a^r (b^i c^i)^r$ for some $r > 0$. In the same way as in the proof of Lemma 3.2, we can show that there exists m ($p \geq m \geq 1$, n is divisible by m) such that M' can make the same rewriting on strings of the form $a^n (b^i c^i)^{n+jm} d^{n+1} e^n$ for all integers $j \geq 0$. In a similar way we can show that M' can make the same rewriting on the strings $a^n (b^i c^i)^{n+jm} d^{n+1} e^{n+j_1 m_1}$ for all integers $j_1 \geq 0$ and some m_1, $p \geq m_1 \geq 1$. In particular, $a^n (b^i c^i)^{n+\frac{n}{m} \cdot m} d^{n+1} e^{n+2m_1} = a^n (b^i c^i)^{2n} d^{n+1} e^{n+2m_1} \notin L_d^{(i)}$ and the 'pumped' cycle yields $a^{n-r} (b^i c^i)^{2n-r} d^{n+1} e^{n+2m_1} \in L_d^{(i)}$, which contradicts the error preserving property (Lem. 2.2).

Thus, M' can rewrite only on the border of d's and e's in C. Because M' is a $\mathsf{w}(i-1)\mathsf{mon\text{-}RRW}$-automaton, in any subsequent cycle the place of rewriting can move to the left by at most $i-1$ positions with respect to the previous cycle. M' must in any case check that the number of a's equals the number of (b^i)'s. This can be done only by rewriting (deleting) on the border of a's and b^i. When the places of rewriting move from the right end of the part containing $b^i c^i$ we can apply the same arguments as in the proof of Lemma 3.2 to show that $\mathsf{w}(i-1)$-monotonicity is not sufficient. □

From this result we immediately obtain the following infinite hierarchies.

Corollary 3.7. *For each $i \geq 0$ and for each $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}\}$,*

(a) $\mathcal{L}(\mathsf{det\text{-}w}(i)\mathsf{mon\text{-}X}) \subset \mathcal{L}(\mathsf{det\text{-}w}(i+1)\mathsf{mon\text{-}X})$.

(b) $\mathcal{L}(\mathsf{w}(i)\mathsf{mon\text{-}X}) \subset \mathcal{L}(\mathsf{w}(i+1)\mathsf{mon\text{-}X})$.

4. Separating the various hierarchies

In this section we will see that the hierarchies of Corollary 3.7 differ pairwise from each other. All the inclusion relations that we will prove are depicted in Figure 1. The vertical arrows and the arrows originating from DCFL are consequences of Corollary 3.7. The horizontal arrows will follow from Theorem 4.3, the left slanted arrows will be derived in Theorems 4.5 and 4.1 (a,c), and the right slanted arrows will be obtained from Theorems 4.7 and 4.1 (b,d).

In [3] Lemmas 4.1 and 4.2, it is shown that

$$\mathcal{L}(\mathsf{mon\text{-}RR}) \setminus \mathcal{L}(\mathsf{RW}) \neq \emptyset \quad \text{and} \quad \mathcal{L}(\mathsf{mon\text{-}RW}) \setminus \mathcal{L}(\mathsf{RR}) \neq \emptyset,$$

which yields the following consequences.

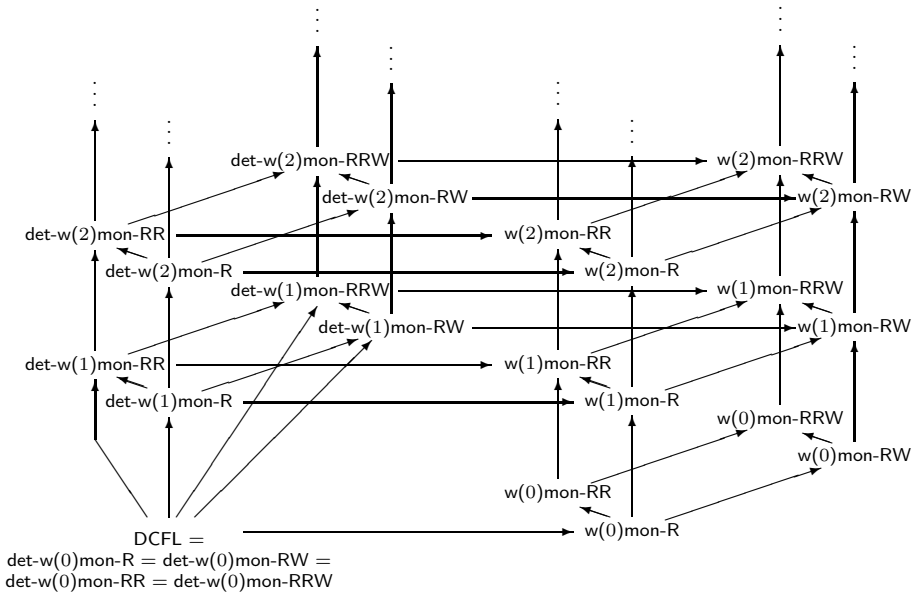

FIGURE 1. Hierarchies of language classes defined by the various types of weakly monotone restarting automata without auxiliary symbols. An arrow $A \longrightarrow B$ indicates that the class $\mathcal{L}(A)$ is strictly included in the class $\mathcal{L}(B)$. When there is no oriented path between any two types of restarting automata in the diagram, then the corresponding language classes are incomparable under inclusion.

Theorem 4.1. *For all* $i \geq 0$,

$$\begin{array}{lrcl} \text{(a)} & \mathcal{L}(\mathsf{w}(i)\mathsf{mon\text{-}RR}) \setminus \mathcal{L}(\mathsf{R}) & \neq & \emptyset. \\ \text{(b)} & \mathcal{L}(\mathsf{w}(i)\mathsf{mon\text{-}RW}) \setminus \mathcal{L}(\mathsf{R}) & \neq & \emptyset. \\ \text{(c)} & \mathcal{L}(\mathsf{w}(i)\mathsf{mon\text{-}RRW}) \setminus \mathcal{L}(\mathsf{RW}) & \neq & \emptyset. \\ \text{(d)} & \mathcal{L}(\mathsf{w}(i)\mathsf{mon\text{-}RRW}) \setminus \mathcal{L}(\mathsf{RR}) & \neq & \emptyset. \end{array}$$

The language $L_{cd} := \{\, c^m d^n \mid 0 \leq m \leq n \leq 2m \,\}$ is easily seen to be accepted by the nondeterministic mon-R-automaton that is given through the following meta-instructions:

1. $(\mathfrak{c}\cdot c^+, cd\cdot d \to d)$; 2. $(\mathfrak{c}\cdot c^+, cdd\cdot d \to d)$; 3. $(\mathfrak{c}\cdot\{cd, cdd, \varepsilon\}\cdot\$, \mathsf{Accept})$.

On the other hand, we have the following negative result.

Lemma 4.2. $L_{cd} \notin \mathcal{L}(\mathsf{det\text{-}RRW})$.

Proof. In order to derive a contradiction, let us assume that L_{cd} is accepted by a det-RRW-automaton M with read/write window of size k. Using pumping techniques we can show that a sufficiently long string $w = c^m d^m$ cannot be accepted

without a restart. After the first cycle of the accepting computation of M on w, we obtain a string $w' = c^{m_1}d^{m_2}$, where $m_1 = m - d_1$ and $m_2 = m - d_2$ for some d_1, d_2 satisfying $k \geq d_1 + d_2 > 0$ and $d_1 \geq d_2$, by the correctness preserving property (Lem. 2.3).

Now M will rewrite in the same way when it starts on the string $c^m d^{2m}$. After this rewriting it cannot reject, because M is deterministic and $c^m d^{2m} \in L_{cd}$. But the resulting string $c^{m-d_1}d^{2m-d_2}$ is not in L_{cd}, as

$$2m - d_2 \geq 2m - d_1 > 2(m - d_1).$$

This violates the correctness preserving property. □

As $L_{cd} \in \mathcal{L}(\mathsf{mon\text{-}R}) \subseteq \mathcal{L}(\mathsf{w}(i)\mathsf{mon\text{-}R})$ for all $i \geq 0$, this yields the following.

Theorem 4.3. *For all $i \geq 0$ and all $X \in \{\mathsf{R}, \mathsf{RW}, \mathsf{RR}, \mathsf{RRW}\}$,*

$$\mathcal{L}(\mathsf{det\text{-}w}(i)\mathsf{mon\text{-}}X) \subset \mathcal{L}(\mathsf{w}(i)\mathsf{mon\text{-}}X).$$

It remains to verify the relations between the different hierarchies of deterministic weakly monotone restarting automata in Figure 1. Recall from [3] that all the considered variants of deterministic monotone restarting automata characterize the language class DCFL.

To this aim we define still another example language $L_b^{(1)} \subset \{a, b, c, d, e, f\}^*$ as $L_b^{(1)} := L_{b1}^{(1)} \cup L_{c,1}^{(1)} \cup L_{c,2}^{(1)}$, where

$$L_{b1}^{(1)} := \left\{ a^i(bc)^j f(bc)^m d \mid i, j, m \geq 0, i = j + m \right\},$$

and $L_{c,1}^{(1)}$ and $L_{c,2}^{(1)}$ are the languages defined in Section 3. Thus, $L_b^{(1)}$ differs from the language $L_c^{(1)}$ by the strings of $L_{b1}^{(1)}$. These additional strings will ensure that, in contrast to $L_c^{(1)}$, $L_b^{(1)}$ is not accepted by any RW-automaton.

A $\mathsf{det\text{-}w}(1)\mathsf{mon\text{-}RR}$-automaton recognizing $L_b^{(1)}$ is obtained by simply adding the meta-instruction $(\mathfrak{c}\cdot a^*\cdot(bc)^*, f \to \varepsilon, (bc)^*\cdot d\$)$ to the instructions of $M_c^{(1)}$ (together with the insertion of appropriate regular languages as the third parts of these meta-instructions). On the other hand, we have the following lemma.

Lemma 4.4. $L_b^{(1)} \notin \mathcal{L}(\mathsf{RW})$.

Proof. Assume to the contrary that some RW-automaton M with read/write window of size k accepts $L_b^{(1)}$. A sufficiently long string $w = a^{2n}(bc)^n f(bc)^n d$ cannot be accepted directly in a tail computation. After the first cycle $q_0\mathfrak{c}w\$ \vdash_M^c q_0\mathfrak{c}w_1\$$ of an accepting computation on w, where q_0 denotes the initial state of M, M must obtain a string w_1 of the form $w_1 = a^{2n}(bc)^{2n}d$ or $w_1 = a^{n+i}(bc)^i f(bc)^n d$ for some $i < n$.

We can assume that i is larger than k and that M did not scan the final symbol d during the cycle $q_0\mathfrak{c}w\$ \vdash_M^c q_0\mathfrak{c}w_1\$$. In the former case let us take the

string $w_2 := a^{2n}(bc)^n f(bc)^n fd \notin L_b^{(1)}$. Then $q_0\¢w_2\$ \vdash_M^c q_0\¢w_3\$$, where $w_3 = a^{2n}(bc)^{2n} fd \in L_b^{(1)}$, which contradicts the error preserving property (Lem. 2.2). In the latter case let us take the string $w_2 := a^{2n}(bc)^{4n}e \notin L_b^{(1)}$. This time $q_0\¢w_2\$ \vdash_M^c q_0\¢w_3\$$, where $w_3 = a^{n+i}(bc)^{i+3n}e \in L_b^{(1)}$, which yields the same contradiction. □

Thus, based on the properties of the language $L_b^{(1)}$ we obtain the following result.

Theorem 4.5. *For all* $i \geq 1$, (a) $\mathcal{L}(\text{det-w}(i)\text{mon-RR}) \setminus \mathcal{L}(\text{R}) \neq \emptyset$.
(b) $\mathcal{L}(\text{det-w}(i)\text{mon-RRW}) \setminus \mathcal{L}(\text{RW}) \neq \emptyset$.

Finally we consider the language $L_{lr} := L_l \cup L_r \subset \{a, l, r\}^*$, where

$$L_l := \left\{ a^{2^n-2i} l a^i \quad \mid n \geq 1,\ 0 \leq i < 2^{n-1} \right\},$$
$$L_r := \left\{ a^i r a^{2^n-2i} \quad \mid n \geq 1,\ 0 \leq i < 2^{n-1} \right\}.$$

Lemma 4.6. $L_{lr} \in \mathcal{L}(\text{det-w}(1)\text{mon-RW}) \setminus \mathcal{L}(\text{RR})$.

Proof. We define an RW-automaton M_{lr} through the following sequence of meta-instructions:

(1) $(\¢\cdot(aa)^+, aal\$ \to la\$)$;
(2) $(\¢\cdot(aa)^+, aala \to laa)$;
(3) $(\¢, aala \to raa)$;
(4) $(\¢\cdot a^*, raaa \to ara)$;
(5) $(\¢\cdot a^+, raa\$ \to al\$)$;
(6) $(\¢\cdot aal\cdot\$, \text{Accept})$;
(7) $(\¢\cdot raa\cdot\$, \text{Accept})$.

It is easily seen that M_{lr} is a det-w(1)mon-RW-automaton that accepts the language L_{lr}. It remains to show that this language is not accepted by any RR-automaton.

Assume to the contrary that there exists an RR-automaton M that accepts L_{lr}, and let k be the size of the read/write window of M. A sufficiently long string $w = a^{2^n} l \in L_{lr}$ cannot be accepted directly by a tail computation. As M cannot use any auxiliary symbols, all the strings that appear on its tape during an accepting computation are accepted by M. Notice further that M can only delete symbols. After the first cycle of an accepting computation on w, M obtains a string w_1 of the form $w_1 = a^{2^n-i} l$, where $0 < i \leq k$, or of the form $w_1 = a^{2^n-i}$, where $0 \leq i < k$. As L_{lr} does not contain any strings of either of these forms, this is the intended contradiction. □

This yields the following theorem completing the results depicted in Figure 1.

Theorem 4.7. *For all* $i \geq 1$, (a) $\mathcal{L}(\text{det-w}(i)\text{mon-RW}) \setminus \mathcal{L}(\text{R}) \neq \emptyset$,
(b) $\mathcal{L}(\text{det-w}(i)\text{mon-RRW}) \setminus \mathcal{L}(\text{RR}) \neq \emptyset$.

5. Conclusions

By considering the degree of weak monotonicity we have obtained infinite hierarchies that lie in the gap between DCFL and CRL in the deterministic case and that lie in the gap between DCFL and GCSL in the nondeterministic case. Naturally, the question arises whether the degree of weak monotonicity induces corresponding hierarchies for the restarting automata with auxiliary symbols. We would certainly expect that. The class $\mathcal{L}(\mathsf{det\text{-}w(1)mon\text{-}RWW})$ is strictly larger than $\mathcal{L}(\mathsf{det\text{-}w(0)mon\text{-}RWW}) = \mathsf{DCFL}$, and also $\mathcal{L}(\mathsf{w(1)mon\text{-}RWW})$ strictly contains $\mathcal{L}(\mathsf{w(0)mon\text{-}RWW}) = \mathsf{CFL}$, because $\mathcal{L}(\mathsf{det\text{-}w(1)mon\text{-}RWW})$ contains non-context-free languages (for example, the language L_{lr} above).

References

[1] E. Dahlhaus and M.K. Warmuth, Membership for growing context-sensitive grammars is polynomial. *J. Comput. Syst. Sci.* **33** (1986) 456–472.

[2] P. Jančar, F. Mráz, M. Plátek and J. Vogel, Restarting automata, in *Proc. FCT'95*, edited by H. Reichel. Springer, Berlin, *Lect. Notes Comput. Sci.* **965** (1995) 283–292.

[3] P. Jančar, F. Mráz, M. Plátek and J. Vogel, On monotonic automata with a restart operation. *J. Autom. Lang. Comb.* **4** (1999) 287–311.

[4] T. Jurdziński, K. Loryś, G. Niemann and F. Otto, Some results on RWW- and RRWW-automata and their relationship to the class of growing context-sensitive languages. Tech. Report 14/01, Fachbereich Mathematik/Informatik, Universität Kassel (2001). Also: To appear in revised form in the *J. Autom. Lang. Comb.*

[5] R. McNaughton, P. Narendran and F. Otto, Church-Rosser Thue systems and formal languages. *J. Assoc. Comput. Mach.* **35** (1988) 324–344.

[6] P. Narendran, *Church-Rosser and related Thue systems*. Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, New York (1984).

[7] G. Niemann and F. Otto, The Church-Rosser languages are the deterministic variants of the growing context-sensitive languages, in *Proc. FoSSaCS'98*, edited by M. Nivat. Springer, Berlin, *Lect. Notes Comput. Sci.* **1378** (1998) 243–257.

[8] G. Niemann and F. Otto, On the power of RRWW-automata, in *Words, Semigroups, and Transductions*, edited by M. Ito, G. Păun and S. Yu. World Scientific, Singapore (2001) 341–355.

[9] G. Niemann and F. Otto, Further results on restarting automata, in *Words, Languages and Combinatorics III, Proc.*, edited by M. Ito and T. Imaoka. World Scientific, Singapore (2003) 352–369.

[10] M. Straňáková, Selected types of pg-ambiguity. *The Prague Bulletin of Mathematical Linguistics* **72** (1999) 29–57.

[11] M. Straňáková, Selected types of pg-ambiguity: Processing based on analysis by reduction, in *Text, Speech and Dialogue, 3rd Int. Workshop, Proc.*, edited by P. Sojka, I. Kopeček and K. Pala. Springer, Berlin, *Lect. Notes Comput. Sci.* **1902** (2000) 139–144.

Communicated by Z. Esik.
Received June 11, 2003. Accepted May 5, 2004.