Acta Informatica (2010) 47:391–412
DOI 10.1007/s00236-010-0125-4

ORIGINAL ARTICLE

# On stateless deterministic restarting automata

**Martin Kutrib · Hartmut Messerschmidt · Friedrich Otto**

Received: 14 January 2010 / Accepted: 16 September 2010 / Published online: 2 October 2010


**Abstract** The transitions of a stateless automaton do not depend on internal states but solely on the symbols currently scanned by its head accessing the input and memory. We investigate stateless deterministic restarting automata that, after executing a rewrite step, continue to read their tape before performing a restart. Even the weakest class thus obtained contains the regular languages properly. The relations between different classes of stateless automata as well as between stateless automata and the corresponding types of automata with states are investigated, and it is shown that the language classes defined by the various types of deterministic stateless restarting automata without auxiliary symbols are anti-AFLs that are not even closed under reversal.

## 1 Introduction

One of the fundamental concepts of computing models is that of internal states that evolve at discrete time steps. At the dawn of automata theory, one of the most important models, the finite automaton, was obtained by extending single Boolean circuits with a finite number of states and feedback [15]. The difference between single Boolean circuits and finite automata is evident. So, the computational power of *stateless* finite automata (actually, *stateless* means

The results of this paper have been announced at SOFSEM 2009 in Špindlerův Mlýn, January 2009. An extended abstract appeared in the proceedings of that conference [10].

M. Kutrib
Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany
e-mail: kutrib@informatik.uni-giessen.de

H. Messerschmidt
Technologie-Zentrum Informatik, Intelligent Systems, 28359 Bremen, Germany
e-mail: hartmut.messerschmidt@tzi.de

F. Otto (✉)
Fachbereich Elektrotechnik/Informatik, Universität Kassel, 34109 Kassel, Germany
e-mail: otto@theory.informatik.uni-kassel.de

that only a single state is present) is strictly weaker than that of general finite automata. On the other hand, it is well-known that stateless nondeterministic pushdown automata already accept all context-free languages, while for deterministic pushdown automata (accepting by empty pushdown) the computational power strictly increases with the number of states [5]. So, in the case of nondeterministic pushdown automata, the resource 'pushdown store' can compensate for the loss of states, while in the case of deterministic pushdown automata, it cannot.

Related studies inspired by biologically motivated models of computing were initiated in [7,16]. Such models as P systems are often stateless, as it is difficult and even unrealistic to maintain a global state for a massively parallel group of objects appearing in natural phenomena of cell evolutions or chemical reactions. The investigation of stateless multihead finite automata and stateless multicounter systems in [16] and the successor papers [3,7] show that the resource 'heads' cannot compensate for the loss of states. Due to the deep relations between certain types of counter machines and P systems [6], further classes of stateless automata deserve to be investigated. Generally speaking, it is a natural and interesting question of how resources given to finite automata relate to the absence or presence of states. Given some computational model, are states necessary at all?

Recently, stateless two-pushdown automata have been investigated [9]. Shrinking as well as length-reducing two-pushdown automata accept exactly the growing context-sensitive languages, and their deterministic counterparts characterize the Church-Rosser languages [1,13]. It turned out that these characterizations remain valid even for the stateless variants of these automata.

Restarting automata were introduced as a formal tool to model *analysis by reduction*, which is a technique used in linguistics to analyze sentences of natural languages [8]. Many restricted types of restarting automata have been studied and put into correspondence with classical families of formal languages (see, e.g., [14] for a recent survey). Restarting automata in connection with descriptional complexity issues are dealt with in [4,12].

In [9] (see also [11]) *stateless restarting automata* were introduced and studied that execute a restart in combination with each rewrite step. It was shown that the expressive power of stateless restarting automata that are deterministic and/or monotone coincides with that of the corresponding types of restarting automata with states, if auxiliary symbols are admitted. However, without auxiliary symbols the stateless types of restarting automata are strictly weaker than the corresponding types with states. But the weakest model, the stateless deterministic and monotone R-automaton (see Sect. 2 for the definitions of the various types of restarting automata), is still sufficiently expressive to accept a superclass of the regular languages.

Here we study the effect that the restriction to a single state has on restarting automata that, after executing a rewrite step, may continue to read their tape before performing a restart. These are the so-called RR-automata and their variants. Thus, even after executing a rewrite operation such an automaton still has the option of accepting or rejecting instead of performing a restart. Actually we will introduce two variants of stateless RR-automata. In one variant the automaton cannot distinguish between the part of a cycle *before* a rewrite step and the part *after* a rewrite step. Such an automaton may try to execute a second rewrite step in a cycle, which is not a legal move. This will then be interpreted as a reject operation. In the second variant the automaton distinguishes between the two parts of a cycle. Correspondingly, it will be called a *stateless two-phase* RR-automaton. We will see that the stateless two-phase RR-automaton can simulate both of the other stateless types of restarting automata. Accordingly in the presence of auxiliary symbols the stateless two-phase restarting

automata that are deterministic and/or monotone are equivalent in expressive power to the corresponding types of restarting automata with states.

We then concentrate on the various types of stateless deterministic restarting automata without auxiliary symbols. We compare the expressive power of these automata to each other, and derive some closure (or rather non-closure) properties for the language classes accepted by these types of automata. As we will see the classes of languages defined by the various types of deterministic stateless restarting automata without auxiliary symbols are anti-AFLs that are not even closed under reversal.

This paper is organized as follows. First we describe in short the basic types of restarting automata and summarize a few fundamental results on them. In the next two sections we present two different types of stateless RRWW-automata, concentrating on their expressive power. Finally in Sect. 5 we present the announced (non-) closure properties. The paper closes with a short summary and some problems for future work.

## 2 Restarting automata

We first describe in short the types of automata we will be dealing with.

An RRWW-*automaton* is a one-tape machine that is described by an 8-tuple

$$M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta),$$

where $Q$ is a finite set of *states*, $\Sigma$ is a finite *input alphabet*, $\Gamma$ is a finite *tape alphabet* containing $\Sigma$, the symbols $¢, \$ \notin \Gamma$ serve as *markers* 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 $\delta$ is the *transition relation* that associates a finite set of *transition steps* to each pair $(q, u)$ consisting of a state $q \in Q$ and a possible content $u$ of the read/write window. There are four types of transition steps:

- A *Move-right step* of the form $(q', \mathsf{MVR})$ causes $M$ to shift the read/write window one position to the right and to enter state $q'$. However, the window cannot be shifted beyond the right border marker \$.
- A *Rewrite step* of the form $(q', v)$, where $q' \in Q$, and $v$ is a string satisfying $|v| < |u|$, causes $M$ to replace the content $u$ of the read/write window by the string $v$, thereby shortening the tape, 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, if $u$ ends with the \$-symbol, then so does $v$, and in this situation the window is placed on the \$-symbol.
- A *Restart step* is of the form Restart. It causes $M$ to place the read/write window over the left end of the tape, so that the first symbol it contains is the left border marker ¢, and to reenter the initial state $q_0$,
- and an *Accept step* is of the form Accept. It causes $M$ to halt and accept.

If $\delta(q, u) = \emptyset$ for some pair $(q, u)$, then $M$ necessarily halts, and we say that $M$ *rejects* in this situation. There is one additional restriction that the transition relation must satisfy: ignoring move operations, *rewrite steps and restart steps alternate* in any computation of $M$, with a rewrite step coming first.

A *configuration* of $M$ is described by a string $\alpha q \beta$, where $q \in Q$, and either $\alpha = \varepsilon$ (the empty word) and $\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{¢\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\$\}$; here $q$ represents the current state, $\alpha\beta$ is the current content of the tape, and it is understood that the head scans

the first $k$ symbols of $\beta$ or all of $\beta$ 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 *cycle* of $M$ is a part of a computation from one restarting configuration to the next. The cycle leading from $q_0 ¢ u \$$ to $q_0 ¢ v \$$ will be denoted as $u \vdash_M^c v$. A computation of $M$ consists of a finite sequence of cycles that is followed by a tail computation, which either accepts or rejects (see, e.g., [14]). A word $w \in \Gamma^*$ is *accepted* by $M$, if there is a computation of $M$ which starts with the configuration $q_0 ¢ w \$$, and which finishes with an accept instruction. By $L_C(M)$ we denote the language consisting of all words over $\Gamma$ that are accepted by $M$. It is the *characteristic language* of $M$, while $L(M) = L_C(M) \cap \Sigma^*$, the set of all input words accepted by $M$, is the (*input*) *language* of $M$. By $S(M)$ we denote the *simple language* of $M$, which consists of all words from $\Sigma^*$ that $M$ accepts in tail computations, that is, in a single sweep without executing a restart operation.

In general, an RRWW-automaton is nondeterministic, that is, to some configurations several different instructions may apply. If that is not the case, then the automaton is called *deterministic*. We use the prefix det- to denote deterministic types of restarting automata. Further types of restarting automata are obtained by combining two other types of restrictions:

(a) Restrictions on the movement of the read/write window (expressed by the first part of the class name): RR-denotes no restriction, and R-means that each rewrite step is combined with a restart. Formally R-automata are obtained from RR-automata by requiring that each rewrite step is immediately followed by a restart step, but it is more convenient to combine the two operations into a single one.

(b) Restrictions on the rewrite-instructions (expressed by the second part of the class name): -WW denotes no restriction, -W means that no auxiliary symbols are available (that is, $\Gamma = \Sigma$), and no W-suffix means that no auxiliary symbols are available and that each rewrite step simply deletes some symbols.

We write $\mathscr{L}(X)$ to denote the *family of languages accepted* by restarting automata of some type $X$. An important property of restarting automata is the following *Correctness Preserving Property*.

**Definition 1** Let $M$ be a restarting automaton. $M$ is *correctness preserving* if, for all words $u$ and $v$ *over its input alphabet*, $u \in L(M)$ and $u \vdash_M^{c^*} v$ imply that $v \in L(M)$, too.

It is easily seen that each deterministic restarting automaton is necessarily correctness preserving, but nondeterministic restarting automata do in general not have this property.

Concerning the expressive power of the various types of restarting automata the following major results have been obtained (see, e.g., [14]). Here a restarting automaton is called *monotone* if the distance of the place of rewriting to the right end of the tape does not increase from one cycle to the next in any computation. We use the prefix mon- to denote monotone types of restarting automata.

**Theorem 1**

$$\begin{array}{lllll}
\text{(a)} & \mathscr{L}(\text{det-mon-R}) & = \mathscr{L}(\text{det-mon-RRWW}) & = \text{DCFL}. \\
\text{(b)} & \mathscr{L}(\text{mon-RWW}) & = \mathscr{L}(\text{mon-RRWW}) & = \text{CFL}. \\
\text{(c)} & \mathscr{L}(\text{det-RWW}) & = \mathscr{L}(\text{det-RRWW}) & = \text{CRL}. \\
\text{(d)} & \mathscr{L}(\text{RRWW}) & \supseteq \mathscr{L}(\text{RWW}) & \supsetneq \text{GCSL}.
\end{array}$$

Here CFL (DCFL) denotes the class of (deterministic) context-free languages, CRL denotes the class of *Church-Rosser languages*, and GCSL is the class of *growing context-sensitive languages* (see, e.g., [1]).

In [9] the *stateless* variants of RWW-automata are studied, where an RWW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ is called *stateless* if $Q = \{q_0\}$ holds. Thus, in this case $M$ can simply be described by the 6-tuple $M = (\Sigma, \Gamma, ¢, \$, k, \delta)$. In the original definition it was required that a stateless RWW-automaton may execute an accept instruction only at the right end of the tape, that is, when it sees the right delimiter \$, but this is actually just a convenience (see [11]). In [9] the following results were obtained on the expressive power of stateless RWW-automata. Here the prefix stl- is used to denote stateless types of restarting automata, and REG denotes the class of regular languages.

**Theorem 2**

(a) $\mathscr{L}$(stl-det-mon-RWW) = DCFL.
(b) $\mathscr{L}$(stl-mon-RWW) = CFL.
(c) $\mathscr{L}$(stl-det-RWW) = CRL.
(d) $\mathscr{L}$(stl-det-mon-R) $\supsetneq$ REG.

## 3 Stateless RR-automata

For restarting automata in general, each RR-variant is at least as powerful as the corresponding R-variant, but for stateless automata the situation is not that obvious. The feature of continuing to read the tape after a rewrite step has been executed is problematic for these automata, as they cannot distinguish between the phase of a cycle *before* the rewrite step and the phase *after* the rewrite step. Clearly, this distinction is important, since no rewrite steps may appear in the latter phase. For general restarting automata, this is avoided by using states, but how to deal with this situation for stateless RR-automata?

We will present two options to deal with this situation in the current and in the next section. In the current section *we regard any additional rewrite step as a rejection of the input*. Also a cycle within which *no* rewrite step is performed *is regarded as a rejection of the input*. The next lemma and example show that for certain languages stateless deterministic R-automata are better suited than even stateless deterministic RRW-automata.

**Lemma 1** *The language* $L_{aba} = \{ a^m b^{m+n} a^n \mid m, n \geq 0 \}$ *is not accepted by any stateless deterministic* RRW*-automaton.*

*Proof* Assume that $M = (\{a, b\}, \{a, b\}, ¢, \$, k, \delta)$ is a stateless deterministic RRW-automaton accepting $L_{aba}$. Then for all integers $m \geq 0$, the words $a^m b^m$ and $b^m a^m$ are all accepted by $M$. For sufficiently large values of $m$, the accepting computation of $M$ on input $a^m b^m$ cannot simply be an accepting tail. Thus, it begins with a cycle of the form $a^m b^m \vdash_M^c w_1$. By the correctness preserving property for $M$, it follows that $w_1$ is an element of $L(M) = L_{aba}$, which means that $w_1$ is of the form $w_1 = a^l b^{l+n} a^n$. However, it is not possible to rewrite $w = a^m b^m$ into a word of this form unless $n = 0$. Therefore, $w_1 = a^{m-i} b^{m-i}$ for some positive integer $i$ satisfying $2i \leq k$. Analogously, the accepting computation of $M$ on input $b^m a^m$ begins with a cycle of the form $b^m a^m \vdash_M^c w_2$, where $w_2$ is of the form $w_2 = b^{m-j} a^{m-j}$ for some positive integer $j$ satisfying $2j \leq k$. Thus, $\delta$ contains rewrite operations of the form $\delta(a^l b^{k-l}) = a^{l-i} b^{k-l-i}$ and $\delta(b^{l'} a^{k-l'}) = b^{l'-j} a^{k-l'-j}$. In addition, $M$ must be able to move its read/write window across the prefix $¢ \cdot a^m$ and across the prefix $¢ \cdot b^m$, that is, $\delta$ must contain the corresponding move-right operations.

Now consider the computation of $M$ on input $a^m b^{2m} a^m \in L_{aba}$. First $M$ will behave just as it does on the input $a^m b^m$, that is, it will rewrite the prefix $a^m b^m$ into $a^{m-i} b^{m-i}$,

which results in the word $a^{m-i}b^{m-i}b^m a^m$, with the read/write window inside the block of $b$'s. Then it will move the window to the right until it contains the factor $b^{l'}a^{k-l'}$, which will then initiate a second rewrite step within the first cycle, that is, $M$ will perform two rewrite operations in *one* cycle. Accordingly, it will reject on input $a^m b^{2m} a^m$. Hence, it follows that $L(M) \neq L_{aba}$, that is, the language $L_{aba}$ is not accepted by any stateless deterministic RRW-automaton. □

However, the language $L_{aba}$ is accepted by a stateless deterministic R-automaton as shown by the following example.

*Example 1* Let $M = (\{a, b\}, \{a, b\}, ¢, \$, 4, \delta)$ be the stl-det-R-automaton that is specified through the following transition function $\delta$:

| | | |
|---|---|---|
| (1) $\delta(¢ \cdot \$) = \mathsf{Accept}$, | (6) $\delta(aaaa) = \mathsf{MVR}$, | (11) $\delta(¢ \cdot bba) = \mathsf{MVR}$, |
| (2) $\delta(¢ \cdot ab \cdot \$) = \mathsf{Accept}$, | (7) $\delta(aaab) = \mathsf{MVR}$, | (12) $\delta(bbbb) = \mathsf{MVR}$, |
| (3) $\delta(¢ \cdot ba \cdot \$) = \mathsf{Accept}$, | (8) $\delta(aabb) = ab$, | (13) $\delta(bbba) = \mathsf{MVR}$, |
| (4) $\delta(¢ \cdot aaa) = \mathsf{MVR}$, | (9) $\delta(¢ \cdot abb) = ¢ \cdot b$, | (14) $\delta(bbaa) = ba$. |
| (5) $\delta(¢ \cdot aab) = \mathsf{MVR}$, | (10) $\delta(¢ \cdot bbb) = \mathsf{MVR}$, | |

The empty word and the words $ab$ and $ba$ are accepted immediately. For input $w = a^2b^3a \in L_{aba}$, $M$ proceeds as follows, where $\vdash_M$ denotes the single-step computation relation that $M$ induces on the set of configurations. Here $q_0$ denotes the unique state of $M$, and we underline the part of the tape contents that is inside $M$'s window:

$$q_0¢\underline{aabbba}\$ \vdash_M ¢q_0\underline{aabbba}\$ \vdash_M q_0¢\underline{abba}\$ \vdash_M q_0¢\underline{ba\$} \vdash_M \mathsf{Accept}.$$

In general, a word of the form $a^m b^{m+n} b^n$ is first reduced to $b^n a^n$, which is then reduced to $ba$ and accepted. On the other hand, if the given input is not of this form, then $M$ will eventually detect that and reject. It follows that $L(M) = L_{aba}$.

Thus, there are languages in $\mathscr{L}$(stl-det-R) that are not accepted by any stateless deterministic RRW-automaton. So, the question for the computational power of stateless RR-automata arises immediately. By a slight modification of the proof of the corresponding result from [9] it can be shown that at least each regular language is accepted by some stl-det-RR-automaton.

**Lemma 2** *Each regular language is accepted by a stateless deterministic* RR*-automaton that is monotone.*

*Proof* Let $L \subseteq \Sigma^*$ be a regular language. Then there exists a complete deterministic finite-state acceptor $A = (Q, \Sigma, q_0, F, \phi)$ for $L$. Let $m := |Q|$. Then each word $w \in \Sigma^m$ can be written as $w = w_1w_2w_3$ such that $|w_2| \geq 1$ and $\phi(q_0, w_1w_2) = \phi(q_0, w_1)$. Hence, $\phi(q_0, w) = \phi(q_0, w_1w_2w_3) = \phi(q_0, w_1w_3)$, which implies that, for all $z \in \Sigma^*$, $wz \in L$ if and only if $w_1w_3z \in L$. In fact, for each word $w \in \Sigma^m$, we can fix one such factorization.

Based on this observation we define a stateless RR-automaton with tape alphabet $\Sigma$ and window of size $k := m + 1$ as follows, where $\Sigma^{<m} = \{ w \in \Sigma^* \mid |w| < m \}$ and $\Sigma^{\leq m} = \{ w \in \Sigma^* \mid |w| \leq m \}$:

| | | |
|---|---|---|
| (1) $\delta(¢ \cdot w \cdot \$) = \mathsf{Accept}$ | for all | $w \in \Sigma^{<m} \cap L$, |
| (2) $\delta(¢ \cdot w) = ¢ \cdot w_1w_3$ | for all | $w \in \Sigma^m$, where $w = w_1w_2w_3$ is the chosen factorization, |
| (3) $\delta(w) = \mathsf{MVR}$ | for all | $w \in \Sigma^{m+1}$, |
| (4) $\delta(w \cdot \$) = \mathsf{Restart}$ | for all | $w \in \Sigma^{\leq m}$. |

Then $M$ is a stateless deterministic RR-automaton, it is obviously monotone, as each rewrite step is applied on a prefix of the current tape contents, and from the choice of the

factorizations it follows that each cycle $u \vdash_M^c u'$ satisfies the property that $u \in L$ if and only if $u' \in L$. Hence, $L(M) = L$. □

However, stateless deterministic RR-automata can also accept some non-regular languages.

*Example 2* The stl-det-RR-automaton $M$ on $\Sigma = \{a, b\}$ that is given through the following transition function $\delta$ is easily seen to accept the non-regular language $\{ a^n b^n \mid n \geq 0 \}$:

$$\begin{array}{lll}
(1)\ \delta(\mathrm{¢} \cdot \$) = \mathsf{Accept} & (5)\ \delta(aaaa) = \mathsf{MVR}, & (9)\ \delta(bbb \cdot \$) = \mathsf{Restart}, \\
(2)\ \delta(\mathrm{¢} \cdot ab \cdot \$) = \mathsf{Accept}, & (6)\ \delta(aaab) = \mathsf{MVR}, & (10)\ \delta(bb \cdot \$) = \mathsf{Restart}, \\
(3)\ \delta(\mathrm{¢} \cdot aaa) = \mathsf{MVR}, & (7)\ \delta(aabb) = ab, & (11)\ \delta(b \cdot \$) = \mathsf{Restart}, \\
(4)\ \delta(\mathrm{¢} \cdot aab) = \mathsf{MVR}, & (8)\ \delta(bbbb) = \mathsf{Restart}, & (12)\ \delta(\$) = \mathsf{Restart}.
\end{array}$$

As the stateless RR-automaton $M$ in the above example is also monotone, we have the following proper inclusion.

**Corollary 1** $\mathsf{REG} \subsetneq \mathscr{L}(\text{stl-det-mon-RR})$.

Currently the exact relationship between the class of languages that are accepted by stl-det-R(W)-automata and those that are accepted by stl-det-RR(W)-automata remains open. Observe, however, that the following characterizations follow in the same way as the corresponding results for stateless RWW-automata, as the proofs given in [11] easily extend to stateless RRWW-automata.

**Theorem 3**

$$\begin{array}{ll}
\text{(a)}\ \mathscr{L}(\text{stl-det-mon-RRWW}) & = \mathsf{DCFL}. \\
\text{(b)}\ \mathscr{L}(\text{stl-mon-RRWW}) & = \mathsf{CFL}. \\
\text{(c)}\ \mathscr{L}(\text{stl-det-RRWW}) & = \mathsf{CRL}.
\end{array}$$

On the other hand, we see below that simple languages of deterministic stateless RR-automata are not necessarily strictly locally testable in contrast to the situation for stateless R(W)(W)-automata [9].

*Example 3* Let $M = (\{a, b\}, \{a, b\}, ¢, \$, 2, \delta)$ be the stateless deterministic RR-automaton that is given through the following transition function $\delta$:

$$\begin{array}{ll}
(1)\ \delta(\mathrm{¢} \cdot a) = \mathsf{MVR}, & (3)\ \delta(ab) = a, \\
(2)\ \delta(aa) = \mathsf{MVR}, & (4)\ \delta(a \cdot \$) = \mathsf{Accept}.
\end{array}$$

Then $L(M) = S(M) = a^+ \cup a^+ \cdot b \cdot a^+$, which is not strictly locally testable. Observe that on input $w \in \{a, b\}^+$ satisfying $|w|_b \geq 2$, $M$ will either get stuck (and so reject), or it will perform two rewrite steps successively (and so reject as well).

Thus, stateless (deterministic) RR(W)(W)-automata are more expressive than stateless (deterministic) R(W)(W)-automata with respect to the simple languages they accept.

In [11] it is shown that we may require that a stateless RWW-automaton executes accept instructions only at the right end of its tape. Does a corresponding normalization result also hold for stateless RRWW-automata? The same question can be stated for the position of restart operations. However, here we have the following negative results.

**Lemma 3** (a) *Stateless deterministic* RRW*-automata are strictly less expressive when they are required to perform accept operations only at the right end of the tape.*

(b) *Stateless deterministic* RRW*-automata are strictly less expressive when they are required to perform restart operations only at the right end of the tape.*

*Proof* (a) Let $\Sigma = \{a, b, c\}$, let $L_{acc} = \{\, a^n b^n \mid n \geq 1 \,\} \cup \{\, a^m cu \mid m \geq 0 \text{ and } u \in \Sigma^* \,\}$, and let $M = (\Sigma, \Sigma, ¢, \$, 4, \delta)$ be the stateless deterministic RR-automaton that is specified by the following transition function $\delta$:

$$
\begin{array}{lll}
(1)\ \delta(¢ \cdot u \cdot \$) & = \mathsf{Accept} & \text{for all } u \in \{ab, ac, c, ca, cb, cc\},\\
(2)\ \delta(¢ \cdot u) & = \mathsf{Accept} & \text{for all } u \in \{a^2c\} \cup ac \cdot \Sigma \cup c \cdot \Sigma^2,\\
(3)\ \delta(¢ \cdot u) & = \mathsf{MVR} & \text{for all } u \in \{a^3, a^2b\},\\
(4)\ \delta(u) & = \mathsf{MVR} & \text{for all } u \in \{a^4, a^3b\},\\
(5)\ \delta(a^2b^2) & = ab, & \\
(6)\ \delta(b^4) & = \mathsf{MVR}, & \\
(7)\ \delta(b^r \cdot \$) & = \mathsf{Restart} & \text{for all } r \in \{0, 1, 2, 3\},\\
(8)\ \delta(a^3c) & = \mathsf{Accept}. &
\end{array}
$$

Obviously, $M$ accepts each word $w \in L_{acc}$, $|w| \leq 2$, immediately by using transition (1). Further, an input $w = a^n b^n$, $n \geq 2$, is reduced to $a^{n-1}b^{n-1}$ by transitions (3) to (7), and so it is eventually reduced to $ab$ and accepted. Finally, an input of the form $w = a^m cu$ is accepted as soon as the prefix $a^m c$ has been read. On the other hand, if a word $w \in \Sigma^*$ is accepted by $M$, then it either has a prefix of the form $a^m c$, or it is of the form $a^n b^n$ for some $n \geq 1$. Thus, we see that $L(M) = L_{acc}$ holds.

Now let $M'$ be a stateless deterministic RRW-automaton of window size $k$ that is required to perform accept instructions only at the right end of the tape. Consider an input of the form $w_1 = a^n b^n$ for a large integer $n$. Clearly, $M'$ cannot accept $w$ in a tail computation, since then it would also accept the word $a^n b^{n+1}$ that does not belong to $L_{acc}$. Thus, the accepting computation of $M'$ on input $w_1$ begins with a cycle of the form $w_1 \vdash^c_{M'} w_2$, where $w_2 \in L_{acc}$. If $w_2$ is of the form $a^m cu$, then $M'$ must execute the above rewrite step within the prefix of length $n + k$ of $w_1$. Accordingly, starting with input $a^n b^{n+1}$, $M'$ would execute the cycle $a^n b^{n+1} \vdash^c_{M'} a^m cub$. As the latter word belongs to the language $L_{acc}$, it follows that $M'$ accepts the word $a^n b^{n+1} \notin L_{acc}$, that is, $L(M') \neq L_{acc}$. Hence, we see that $w_2$ is necessarily of the form $w_2 = a^{n-s} b^{n-s}$, that is, $M'$ executes the rewrite step $\delta'(a^r b^{k-r}) = a^{r-s} b^{k-r-s}$ for some $s \leq r \leq k - s$.

Next consider the input $w_3 = a^n c a^n b^{k-r} a^n b^n \in L_{acc}$. As $M'$ can execute accept instructions only at the right end of the tape, it either just moves right across the symbol $c$, or it applies a rewrite operation with the symbol $c$ inside its window. In the former case $M'$ reaches the configuration $a^n c \cdot a^k \cdot a^{n-k} b^{k-r} a^n b^n$ with the window on the factor $a^k$ displayed. As $M'$ is stateless, it will now behave as on input $w_1$, that is, it will move right until its window contains the factor $a^r b^{k-r}$, which it will then rewrite into $a^{r-s} b^{k-r-s}$. The resulting configuration is $a^n c a^{n-s} b^{k-r-s} \cdot a^k \cdot a^{n-k} b^n$ with the window on the factor $a^k$ displayed. Still $M'$ will keep on moving to the right, and accordingly, it will execute the same rewrite operation again on the last factor $a^n b^n$, in this way rejecting input $w_3$.

If $M'$ executes a rewrite operation on encountering the symbol $c$ in $w_3$, then this rewrite operation is of the form $\delta'(a^i c a^j) = v$ for some integers $i, j \geq 0$ satisfying $k = i + j + 1$. Thus, the resulting configuration will be of the form $a^{n-i} v \cdot a^k \cdot a^{n-k-j} b^{k-r} a^n b^n$. As above $M'$ will now behave as on input $w_1$, that is, it will execute the rewrite operation $\delta'(a^r b^{k-r}) = a^{r-s} b^{k-r-s}$ on the factor $a^{n-k-j} b^{k-r}$, in this way rejecting input $w_3$. It follows in either case that $L(M') \neq L_{acc}$ holds.

(b) Let $\Sigma = \{a, b, c\}$, let $L_{rs} = \{\, a^n b^n c \mid n \geq 1 \,\} \cdot \{a, b, c\}^*$, and let $M = (\Sigma, \Sigma, ¢, \$, 5, \delta)$ be the stateless deterministic RR-automaton that is specified by the

following transition function $\delta$:

$$
\begin{array}{lll}
(1)\ \delta(\mathfrak{c}\cdot u) & = \mathsf{MVR} & \text{for all } u \in \{a^4, a^3b\},\\
(2)\ \delta(\mathfrak{c}\cdot a^2b^2) & = \mathfrak{c}\cdot ab, & \\
(3)\ \delta(u) & = \mathsf{MVR} & \text{for all } u \in \{a^5, a^4b\},\\
(4)\ \delta(a^3b^2) & = a^2b, & \\
(5)\ \delta(u) & = \mathsf{Restart} & \text{for all } u \in \{b,c\}\cdot(\Sigma^4 \cup \Sigma^{\leq 3}\cdot\$),\\
(6)\ \delta(\mathfrak{c}\cdot abcx) & = \mathsf{Accept} & \text{for all } x \in \Sigma,\\
(7)\ \delta(\mathfrak{c}\cdot abc\cdot\$) & = \mathsf{Accept}. &
\end{array}
$$

It is easily seen that $L(M) = L_{rs}$ holds.

On the other hand, let $M'$ be a stateless deterministic RRW-automaton that is required to perform restart operations only at the right end of the tape. First consider the input $w_1 = a^nb^nc$, where $n$ is a large positive integer. Clearly, $M'$ cannot accept $w_1$ in a tail computation, since then it would also accept the word $a^nb^{n+1}c$ that does not belong to $L_{rs}$. Thus, the accepting computation of $M'$ on input $w_1$ begins with a cycle of the form $w_1 \vdash^c_{M'} w_2$, where $w_2 \in L_{rs}$. Hence, we see that $w_2$ is necessarily of the form $w_2 = a^{n-s}b^{n-s}c$, that is, $M'$ executes the rewrite step $\delta'(a^rb^{k-r}) = a^{r-s}b^{k-r-s}$ for some $s \leq r \leq k-s$. Now consider the input $w_3 = a^nb^nca^nb^n \in L_{rs}$. Again $M'$ cannot accept $w_3$ in a tail computation, that is, the accepting computation of $M'$ on input $w_3$ begins with a cycle of the form $w_3 \vdash^c_{M'} w_4$. As $M'$ is deterministic, we see that within this cycle, the prefix $a^nb^n$ of $w_3$ is rewritten into $a^{n-s}b^{n-s}$, that is, $w_4 = a^{n-s}b^{n-s}ca^nb^n$. After performing this rewrite operation, $M'$ needs to make a restart. By assumption it can execute a restart operation only at the right end of the tape, and so it must move its read/write window across the suffix $b^{n-k+r}ca^nb^n$ of $w_4$. This, however, means that it will perform another rewrite step on the suffix $a^nb^n$, in this way rejecting input $w_3$. It follows that $L(M') \neq L_{rs}$, that is, the language $L_{rs}$ is not accepted by any stateless deterministic RRW-automaton that is required to execute restart steps only at the right end of the tape. □

## 4 Stateless two-phase RR-automata

The above problems with the stateless RRWW-automaton are a consequence of the fact that such an automaton is unable to remember whether or not it has already executed a rewrite operation in the current cycle. Here we consider a variant of stateless RRWW-automata that can distinguish between these two cases.

A *stateless two-phase* RRWW*-automaton*, stl-2-RRWW-automaton for short, is described by a 7-tuple $M = (\Sigma, \Gamma, \mathfrak{c}, \$, k, \delta_1, \delta_2)$, where $\delta_1$ is the transition relation that specifies the behavior of $M$ during the first phase of each cycle, which ends with the execution of a rewrite instruction, while $\delta_2$ is the transition relation that specifies the behavior of $M$ after executing a rewrite. Accept instructions can occur in both $\delta_1$ and $\delta_2$. In the former case they correspond to accepting tail computations in which no rewrite step is executed, while in the latter case they correspond to accepting tail computations in which a rewrite step is executed.

For each type $\mathsf{X} \in \{\mathsf{WW}, \mathsf{W}, \varepsilon\}$, we can simulate each stl-RX-automaton and each stl-RRX-automaton cycle by cycle by a stl-2-RRX-automaton. For the former the simulating stl-2-RRX-automaton simply has to restart immediately upon entering the second phase of a cycle. For the latter the $\delta_1$-function of the simulating stl-2-RRX-automaton is obtained from the $\delta$-function of the stl-RRX-automaton being simulated by deleting all restart steps, and its $\delta_2$-function is obtained from $\delta$ by deleting all rewrite steps, in this way turning these rewrite steps into reject operations. Thus, we have the following inclusion results.

**Proposition 1** *For each* $\mathsf{X} \in \{\mathsf{WW}, \mathsf{W}, \varepsilon\}$,

$$\begin{aligned} &\text{(a) } \mathscr{L}(\mathsf{stl\text{-}(det)\text{-}RX}) \subseteq \mathscr{L}(\mathsf{stl\text{-}(det)\text{-}2\text{-}RRX}).\\ &\text{(b) } \mathscr{L}(\mathsf{stl\text{-}(det)\text{-}RRX}) \subseteq \mathscr{L}(\mathsf{stl\text{-}(det)\text{-}2\text{-}RRX}). \end{aligned}$$

□

This yields the following consequences from Theorem 3.

**Corollary 2**

$$\begin{aligned} &\text{(a) } \mathscr{L}(\mathsf{stl\text{-}det\text{-}mon\text{-}2\text{-}RRWW}) = \mathsf{DCFL}.\\ &\text{(b) } \mathscr{L}(\mathsf{stl\text{-}mon\text{-}2\text{-}RRWW}) = \mathsf{CFL}.\\ &\text{(c) } \mathscr{L}(\mathsf{stl\text{-}det\text{-}2\text{-}RRWW}) = \mathsf{CRL}. \end{aligned}$$

On the other hand, by using essentially the same arguments as in the proof of Lemma 2 of [9], the following negative result can be established.

**Lemma 4** *The deterministic linear context-free language*

$$L_d = \{\, ca^nb^n \mid n \geq 0 \,\} \cup \{\, da^nb^{2n} \mid n \geq 0 \,\}$$

*is not accepted by any stateless two-phase* RRW*-automaton.*

*Proof* Assume that $M = (\Sigma, \Sigma, ¢, \$, k, \delta_1, \delta_2)$ is a stateless 2-RRW-automaton such that $L(M) = L_d$ holds, where $\Sigma := \{a, b, c, d\}$. Then $M$ has an accepting computation for the word $w := da^nb^{2n}$, where $n > k$. Obviously, this computation cannot be an accepting tail, that is, it begins with a cycle of the form $w \vdash_M^c w'$. As we consider an accepting computation, it follows that $w' \in L_d$, which means that $w' = da^{n-i}b^{2n-2i}$ for some $i \leq k/3$. Thus, $M$ contains the rewrite operation $a^{l-i}b^{k-l-2i} \in \delta_1(a^lb^{k-l})$ for some $l \geq i$, and after executing this rewrite operation $M$ performs a restart on the suffix $b^{2n-k+l}\$$ of the tape contents. Next, $M$ also has an accepting computation for the word $z := ca^nb^n$. Finally, consider the word $v := ca^nb^{n+i}$ that does not belong to $L_d$. Starting with input $v$, $M$ cannot simply reject as it cannot distinguish between $v$ and $z$ before it has seen the tape content completely. Thus, the computation of $M$ on input $v$ begins by moving the read/write window of $M$ to the border between the factors $a^n$ and $b^{n+i}$. However, at that place $M$ can apply the above rewrite operation, which yields the configuration $¢ca^{n-i}b^{k-l-2i}q_0b^{n+i-k+l}\$$, where $q_0$ denotes the unique state of $M$. As $M$ is stateless, it cannot distinguish the suffix $b^{n+i-k+l}\$$ from the suffix $b^{2n-k+l}\$$ above, and so it will also perform a restart on the former, thus completing the cycle $v = ca^nb^{n+i} \vdash_M^c ca^{n-i}b^{n-i}$. This, however, means that together with the word $ca^{n-i}b^{n-i} \in L_d$, $M$ will also accept the word $v \notin L_d$, contradicting our assumption that $L(M) = L_d$ holds. Thus, $L_d$ is not accepted by any stateless two-phase RRW-automaton.

□

It follows that without auxiliary symbols, stateless two-phase restarting automata are strictly less expressive than their counterparts with states, that is, $\mathscr{L}(\mathsf{stl\text{-}(det)\text{-}2\text{-}RRX}) \subsetneq \mathscr{L}(\mathsf{(det)\text{-}RRX})$ holds for all $\mathsf{X} \in \{\mathsf{W}, \varepsilon\}$. In the sequel we will repeatedly utilize the witness language $L_{\text{expo}}^{(1)}$:

$$L_{\text{expo}}^{(1)} = \{\, a^{2^n} \mid n \geq 0 \,\} \cup \{\, a^iba^j \mid i, j \geq 0, \text{ and } \exists m \geq 1 : i + 2 \cdot j = 2^m \,\}.$$

**Lemma 5** $L_{\text{expo}}^{(1)} \in \mathscr{L}(\mathsf{stl\text{-}det\text{-}2\text{-}RRW})$.

*Proof* Let $M = (\{a, b\}, \{a, b\}, ¢, \$, 5, \delta_1, \delta_2)$ be the stateless deterministic two-phase RRW-automaton that is specified through the following transition functions:

$$\begin{array}{lll}
(1)\ \delta_1(¢ \cdot x \cdot \$) = \mathsf{Accept}, \text{ for all } x \in \{a, a^2, ba, a^2b, ba^2\}, & & \\
(2)\ \delta_1(¢ \cdot a^4) = \mathsf{MVR}, & (6)\ \delta_1(a^4b) = ba^2, & \\
(3)\ \delta_1(¢ \cdot a^2ba) = ¢ \cdot a^2, & (7)\ \delta_1(a^4 \cdot \$) = ba^2 \cdot \$, & \\
(4)\ \delta_1(¢ \cdot ba^3) = ¢ \cdot a^3, & (8)\ \delta_2(a^5) = \mathsf{MVR}, & \\
(5)\ \delta_1(a^5) = \mathsf{MVR}, & (9)\ \delta_2(a^i \cdot \$) = \mathsf{Restart}, \text{ for all } 0 \le i \le 4. &
\end{array}$$

Let $w \in \{a, b\}^*$. If $|w| \le 4$, then it is easily seen that $M$ will accept on input $w$ if and only if $w \in L^{(1)}_{\text{expo}}$ holds. Thus, assume that $|w| \ge 5$. If $|w|_b \ge 2$, then $M$ will get stuck on reading $w$. If $w = a^iba^j$, then $M$ rejects immediately if $i = 1$ or if $i = 3$. If $i = 0$ or $i = 2$, then $M$ immediately executes a rewrite operation that transforms $w$ into the word $w_1 = a^{i/2+j}$ and restarts, and if $i \ge 4$, then it rewrites $w$ into the word $w_1 = a^{i-4}ba^{j+2}$ and restarts. Finally, if $w = a^m$, then $w$ is rewritten into $w_1 = a^{m-4}ba^2$. Thus, in each case $w_1$ belongs to $L^{(1)}_{\text{expo}}$ if and only if $w$ does. It follows that $L(M) = L^{(1)}_{\text{expo}}$ holds. □

Next we will see that stateless RW- and RRW-automata are less expressive.

**Lemma 6** $L^{(1)}_{\text{expo}} \notin \mathscr{L}(\text{stl-RW}) \cup \mathscr{L}(\text{stl-det-RRW})$.

*Proof* Assume that $M = (\{a, b\}, \{a, b\}, ¢, \$, k, \delta)$ is a stateless RW-automaton such that $L^{(1)}_{\text{expo}} = L(M)$ holds. Consider the word $w = ba^{2^n} \in L^{(1)}_{\text{expo}}$, where $n$ is sufficiently large. If $M$ accepts $w$ in a tail computation, then it will also accept the word $wa = ba^{2^n+1} \notin L^{(1)}_{\text{expo}}$. On the other hand, if an accepting computation of $M$ on input $w$ begins with a cycle of the form $w \vdash^c_M w_1$, then $w_1$ must be an element of $L^{(1)}_{\text{expo}}$. As each rewrite step of $M$ is length-reducing, it follows that $w_1 = a^{2^n}$, that is, $M$ rewrites the prefix $¢ \cdot ba^{k-1}$ into the word $¢ \cdot a^{k-1}$. Now consider the word $z = ba^{2^n}ba^{2^{n-1}} \notin L^{(1)}_{\text{expo}}$. On input $z$, $M$ can execute the cycle $z = ba^{2^n}ba^{2^{n-1}} \vdash^c_M a^{2^n}ba^{2^{n-1}} \in L^{(1)}_{\text{expo}}$, which implies that $z \in L(M)$. It follows that $L(M) \ne L^{(1)}_{\text{expo}}$.

Now assume that $M' = (\{a, b\}, \{a, b\}, ¢, \$, k, \delta)$ is an stl-det-RRW-automaton such that $L^{(1)}_{\text{expo}} = L(M')$ holds. Consider the word $w = ba^{2^n} \in L^{(1)}_{\text{expo}}$, where $n$ is sufficiently large. As above it follows that the accepting computation of $M'$ on input $w$ begins with a cycle of the form $w \vdash^c_{M'} w_1$, which means that $w_1 \in L^{(1)}_{\text{expo}}$. As each rewrite step of $M'$ is length-reducing, it follows that $w_1 = a^{2^n}$, that is, $M'$ rewrites the prefix $¢ \cdot ba^{k-1}$ into the word $¢ \cdot a^{k-1}$. Further, this cycle ends with a restart operation of the form $\delta(a^k) = \mathsf{Restart}$ or of the form $\delta(a^j \cdot \$) = \mathsf{Restart}$ for some $j \le k - 1$. In the former case $M'$ can execute the cycle $ba^{2^n}ba^{2^{n-1}} \vdash^c_{M'} a^{2^n}ba^{2^{n-1}} \in L^{(1)}_{\text{expo}}$, which would imply that $M'$ accepts on input $ba^{2^n}ba^{2^{n-1}}$, although $ba^{2^n}ba^{2^{n-1}} \notin L^{(1)}_{\text{expo}}$. Thus, a restart operation is executed at the right delimiter \$, that is, $M'$ moves its read/write window across the suffix $a^{2^n-k+1}$ of the word $w$ and performs a restart operation on encountering the symbol \$.

Next consider the word $w' = a^{2^n} \in L^{(1)}_{\text{expo}}$. Again, if $M'$ accepts $w'$ in a tail computation, then it will also accept the word $w'a = a^{2^n+1} \notin L^{(1)}_{\text{expo}}$. Thus, the accepting computation of $M'$ on input $w'$ begins with a cycle $w' \vdash^c_{M'} w'_1$, where $w'_1 \in L^{(1)}_{\text{expo}}$. It follows that $w'_1 = a^{2^n-2i}ba^i$ holds for some $i$, $2 \le i < k/2$, that is, $M'$ rewrites the suffix $a^{k-1} \cdot \$$ into the word $a^{k-1-2i}ba^i \cdot \$$ and performs a restart afterwards. Thus, on the suffix $a^{2^n-k+1} \cdot \$$ of $w'$, $M'$ has two possible sequences of steps: it can either rewrite $a^{k-1} \cdot \$$ into $a^{k-1-2i}ba^i \cdot \$$

and restart, or it can just move right across the suffix $a^{2^n-k+1}$ and restart on reaching the \$-symbol without performing a rewrite step (see above). This, however, contradicts our assumption that $M'$ is deterministic. It follows that $L(M') \neq L^{(1)}_{\text{expo}}$. □

Together with Proposition 1, Lemmas 5 and 6 yield the following proper inclusions.

**Corollary 3**

(a) $\mathscr{L}(\text{stl-det-RW}) \subsetneq \mathscr{L}(\text{stl-det-2-RRW})$.
(b) $\mathscr{L}(\text{stl-det-RRW}) \subsetneq \mathscr{L}(\text{stl-det-2-RRW})$.

To obtain corresponding results for stl-det-2-RR-automata, we consider the following example language.

**Definition 2** Let $\phi : \{a, b\}^* \to \{a, b\}^*$ be the morphism that is induced by mapping $a \mapsto ab$ and $b \mapsto b$. Then $\phi$ is an injective mapping, that is, it is an encoding. Now let $L^{(1)}_\phi$ be the language $L^{(1)}_\phi = \phi(L^{(1)}_{\text{expo}})$, that is,

$$L^{(1)}_\phi = \left\{ (ab)^{2^n} \mid n \geq 0 \right\} \cup \left\{ (ab)^i b (ab)^j \mid i, j \geq 0, \text{ and } \exists m \geq 1 : i + 2 \cdot j = 2^m \right\}.$$

Then the following result holds.

**Lemma 7** $L^{(1)}_\phi \in \mathscr{L}(\text{stl-det-2-RR})$.

*Proof* Let $M = (\{a, b\}, \{a, b\}, ¢, \$, 9, \delta_1, \delta_2)$ be the stateless deterministic two-phase RR-automaton that is specified through the following transition functions:

(1) $\delta_1(¢ \cdot x \cdot \$) = \text{Accept}$, for $x \in \{ab, (ab)^2, b(ab), b(ab)^2, (ab)^2 b, (ab)^2 b(ab)\}$,
(2) $\delta_1(¢ \cdot (ab)^4) = \text{MVR}$,
(3) $\delta_1((ab)^3 b(ab)) = \text{MVR}$,
(4) $\delta_1((ab)^4 a) = \text{MVR}$,
(5) $\delta_1(b(ab)^4) = \text{MVR}$,
(6) $\delta_1((ab)^4 b) = \text{MVR}$,
(7) $\delta_1(b(ab)^3 ba) = \text{MVR}$,
(8) $\delta_1((ab)^4 \cdot \$) = b(ab)^2 \cdot \$$,
(9) $\delta_1(¢ \cdot b(ab)^3 a) = ¢ \cdot (ab)^3 a$,
(10) $\delta_1(¢ \cdot (ab)^2 b(ab) a) = ¢ \cdot (ab)^2 a$,
(11) $\delta_1(b(ab)^2 b(ab) a) = bb(ab)^2 a$,
(12) $\delta_2((ab)^4 a) = \text{MVR}$,
(13) $\delta_2(b(ab)^4) = \text{MVR}$,
(14) $\delta_2(b(ab)^i \cdot \$) = \text{Restart}$, $0 \leq i \leq 3$,
(15) $\delta_2((ab)^4 \cdot \$) = \text{Restart}$.

Essentially $M$ simulates the stateless deterministic two-phase RRW-automaton from the proof of Lemma 5. Observe that the rewritten syllable in rules (9), (10) and (11) always ends with the letter $a$. This implies that $M$ will never accept a word that contains the factor $aa$, as such a factor is not contained in any word accepted in a tail computation (rule (1)), nor can $M$ ever remove such a factor from the tape. Thus, whenever one of the rules (9) to (11) is applied within an accepting computation, then the first letter to the right of the place where the rewriting is executed is necessarily a $b$. Together with the fact that the rewritten syllables always end with the letter $a$, this ensures that $M$ will detect the occurrence of a second copy of $\phi(b)$ in the actual tape content from $\phi(\{a, b\}^*)$. Based on this observation it can now be shown that $L(M) = L^{(1)}_\phi$. □

Using essentially the same arguments as in the proof of Lemma 6 it can be shown that $L^{(1)}_\phi$ is neither accepted by a stateless R- nor by a stateless deterministic RR-automaton, which yields the following separation results.

**Corollary 4**

$$\begin{array}{ll} \text{(a)} & \mathscr{L}(\text{stl-det-R}) \subsetneq \mathscr{L}(\text{stl-det-2-RR}). \\ \text{(b)} & \mathscr{L}(\text{stl-det-RR}) \subsetneq \mathscr{L}(\text{stl-det-2-RR}). \end{array}$$

Finally, we consider the complement $\bar{L}^{(1)}_{\text{expo}} = \{a, b\}^* \setminus L^{(1)}_{\text{expo}}$ of the language $L^{(1)}_{\text{expo}}$. Obviously,

$$\begin{aligned} \bar{L}^{(1)}_{\text{expo}} = &\{ w \in \{a, b\}^* \mid |w|_b \geq 2 \} \cup \{ a^m \mid m \text{ is not a power of } 2 \} \\ &\cup \left\{ a^i b a^j \mid i, j \geq 0, \text{ and } i + 2 \cdot j \text{ is not a power of } 2 \right\}. \end{aligned}$$

It is not hard to construct a stl-det-2-RRW-automaton $\overline{M}$ for this language. On words of the form $a^m$ or $a^i b a^j$, $\overline{M}$ behaves essentially just like the stateless deterministic RRW-automaton $M$ for the language $L^{(1)}_{\text{expo}}$ from the proof of Lemma 5, only that it accepts on other words of length at most 3, while it simply accepts on encountering two $b$'s that are close together in the first phase of a cycle or on encountering a $b$ in the second phase of a cycle. On the other hand, we have the following result.

**Lemma 8** *$\bar{L}^{(1)}_{\text{expo}}$ is not accepted by any stateless deterministic two-phase* RRW*-automaton that executes accept instructions only at the right end of the tape.*

*Proof* Assume that there is a stl-det-2-RRW-automaton $M = (\{a, b\}, \{a, b\}, ¢, \$, k, \delta_1, \delta_2)$ that executes accept instructions only at the right end of the tape and that accepts the language $\bar{L}^{(1)}_{\text{expo}}$.

First consider the input $w_1 = ba^{2^m+1} \in \bar{L}^{(1)}_{\text{expo}}$, where $m > 0$ is a large integer. Given $w_1$ as input, $M$ will accept, but it cannot accept in a tail computation, since then it would also accept the word $ba^{2^m}$ that does not belong to the language $\bar{L}^{(1)}_{\text{expo}}$. Hence, the accepting computation of $M$ on input $w_1$ begins with a cycle of the form $w_1 \vdash^c_M w_2$. In this cycle $M$ executes a rewrite step $\delta_1(u) = v$ such that either $u = ¢ \cdot ba^{k-2}$, or $u = ba^{k-1}$, or $u = a^k$, or $u = a^j \cdot \$$ for some $1 \leq j \leq k-1$. Accordingly, $¢ \cdot w_2 = va^{2^m+3-k}$, or $w_2 = va^{2^m+2-k}$, or $w_2 = bva^{2^m+1-k}$, or $w_2 \cdot \$ = ba^{2^m+1-j}v$. As $M$ is stateless, it will execute a corresponding cycle for the input word $w_1 a^{2^m-1} = ba^{2^{m+1}} \in L^{(1)}_{\text{expo}}$, resulting in the word $¢ \cdot w_3 = va^{2^{m+1}+2-k}$, or $w_3 = va^{2^{m+1}+1-k}$, or $w_3 = bva^{2^{m+1}-k}$, or $w_3 \cdot \$ = ba^{2^{m+1}-j}v$. The correctness preserving property for $M$ requires that $w_2 \in \bar{L}^{(1)}_{\text{expo}}$, while $w_1 a^{2^m-1} \in L^{(1)}_{\text{expo}}$ implies that $w_3 \in L^{(1)}_{\text{expo}}$.

If $|u|_b = 0 = |v|_b$, then the above rewrite operation simply removes $j \geq 1$ occurrences of the symbol $a$, where $j \leq k$. As $m$ is large, this implies that $w_3 \in \bar{L}^{(1)}_{\text{expo}}$, a contradiction. If $|v|_b \geq 2$, or if $|u|_b = 0$ and $|v|_b = 1$, we obtain the same contradiction.

Next consider the case that $|u|_b = 1$ and $|v|_b = 1$. Then the above rewrite operation is of the form $\delta_1(¢ \cdot ba^{k-2}) = ¢ \cdot a^i b a^{k-2-r}$ or $\delta_1(ba^{k-1}) = a^i b a^{k-1-r}$ for some $i, r \geq 0$ satisfying $i < r \leq k-2$ (respectively, $i < r \leq k-1$). However, this implies that $w_3 = a^i b a^{2^{m+1}-r} \in \bar{L}^{(1)}_{\text{expo}}$, again giving the same contradiction.

Finally, assume that $|u|_b = 1$ and $|v|_b = 0$, that is, the rewrite operation above is of the form $\delta_1(¢ \cdot ba^{k-2}) = ¢ \cdot a^{k-2-j}$ or $\delta_1(ba^{k-1}) = a^{k-1-j}$ for some $j \geq 0$. Then $w_3 = a^{2^{m+1}-j}$, which again yields the same contradiction as above if $j > 0$. It follows that $j = 0$. Now consider the input $w_4 = ba^{2^{m+1}} ba^{2^m} \in \bar{L}^{(1)}_{\text{expo}}$. On input $w_4$, $M$ will accept, but the corresponding accepting computation begins with an application of the above rewrite operation. Thus, $w_4 = ba^{2^{m+1}} ba^{2^m}$ is rewritten into the word $a^{2^{m+1}} ba^{2^m} \in L^{(1)}_{\text{expo}}$. After executing this rewrite operation $M$ will scan the remaining tape. On encountering the second occurrence

of the letter $b$, it "knows" that the given input $w_4$ belongs to the language $\bar{L}^{(1)}_{\text{expo}}$, but according to our assumption $M$ cannot accept at that point. Instead it has to scan the remaining suffix $a^{2^m}$. Now on encountering the right delimiter \$, $M$ does not remember that it has seen two occurrences of the letter $b$, that is, it cannot distinguish between $w_4$ and the input $w_5 = ba^{2^m} \notin \bar{L}^{(1)}_{\text{expo}}$. Thus, $M$ either accepts, that is, it accepts both $w_4$ and $w_5$, or it makes a restart, which means that it executes the cycle $w_4 \vdash^c_M a^{2^{m+1}} ba^{2^m} \in L^{(1)}_{\text{expo}}$, contradicting the correctness preserving property for $M$. As this covers all cases, it follows that $L(M) \neq \bar{L}^{(1)}_{\text{expo}}$. □

It can further be shown that the language $\bar{L}^{(1)}_{\phi} = \{a, b\}^* \setminus L^{(1)}_{\phi}$ is not accepted by any stl-det-2-RR-automaton that executes all its accept instructions at the right end of the tape, but that an (unrestricted) stl-det-2-RR-automaton can accept this language. Thus, by restricting stl-det-2-RR(W)-automata to execute accept instructions only at the right end of the tape we decrease their expressive power properly. This contrasts the situation for stateless RW-automata.

## 5 Closure properties

Finally we consider closure properties of those language families that are specified by the various types of stateless deterministic restarting automata without auxiliary symbols. Closure under certain operations indicates a certain robustness of the language families considered, while non-closure properties may serve, for example, as a valuable basis for extensions.

As it turns out all language families considered here form non-reversal closed anti-AFLs, and some of them are not even closed under complementation. This is somewhat surprising for language classes defined by deterministic models of automata. Anti-AFLs are sometimes referred to as "unfortunate families of languages," but there is linguistical evidence that such language families might be of crucial importance, since the family of natural languages is an anti-AFL, too [2]. Hence, the quest for uncommon automata models that induce anti-AFLs seems to be worthwhile. Our results are summarized in Table 1 at the end of the paper.

### 5.1 Boolean operations

First we explore the closure properties under the Boolean operations union, intersection, and complementation. The languages $L_{d,1} = \{ ca^n b^n \mid n \geq 0 \}$ and $L_{d,2} = \{ da^n b^{2n} \mid n \geq 0 \}$ are easily seen to be accepted by stl-det-R- as well as by stl-det-RR-automata. As shown in Lemma 4, their union $L_d = L_{d,1} \cup L_{d,2}$ is not accepted by any stl-det-2-RRW-automaton. This yields the following non-closure results.

**Theorem 4** *The families $\mathscr{L}$(stl-det-R(R)(W)) and $\mathscr{L}$(stl-det-2-RR(W)) are not closed under union.*

Let $M$ be a stateless deterministic R(W)-automaton or a stateless deterministic two-phase RR(W)-automaton accepting a language $L \subseteq \Sigma^*$. Then by switching accepting and rejecting (that is, undefined) transition steps we obtain an automaton of the same type as $M$ that accepts the language $\bar{L} = \Sigma^* \setminus L$. This gives our only closure properties.

**Theorem 5** *The families $\mathscr{L}$(stl-det-R(W)) and $\mathscr{L}$(stl-det-2-RR(W)) are closed under complementation.*

This result contrasts with the following negative result.

**Theorem 6** *The families $\mathscr{L}$(stl-det-RR(W)) are not closed under complementation.*

*Proof* Consider the language $L_r = \{ wcw^R \mid w \in \{a,b\}^* \}$, and let $M_r$ be the stl-det-RR-automaton that is given through the following transition function, where $x \in \{a,b,c\}$ and $y,z \in \{a,b,\varepsilon\}$:

$$\begin{array}{lll} (1)\ \delta(\mathrm{¢}\cdot c\cdot \$) = \mathsf{Accept}, & (5)\ \delta(abx) = \mathsf{MVR}, & (9)\ \ \delta(bcb) \ \ = c, \\ (2)\ \delta(\mathrm{¢}\cdot ax) \ = \mathsf{MVR}, & (6)\ \delta(bax) = \mathsf{MVR}, & (10)\ \delta(yz\cdot \$) = \mathsf{Restart}. \\ (3)\ \delta(\mathrm{¢}\cdot bx) \ = \mathsf{MVR}, & (7)\ \delta(bbx) = \mathsf{MVR}, & \\ (4)\ \delta(aax) \ \ = \mathsf{MVR}, & (8)\ \delta(aca) = c, & \end{array}$$

Given an input $w \in \{a,b,c\}^+$, $M_r$ accepts immediately, if $w = c$. Otherwise it moves right until it discovers a factor of the form $aca$ or $bcb$, which it then replaces by $c$. If no such factor is discovered, then $M_r$ rejects on input $w$, as it either gets stuck on encountering a factor of the form $acb$, $bca$, or $c\cdot\$$, or, if $|w|_c = 0$, then $M_r$ simply moves across $w$ and performs a restart at the right border marker, that is, it makes a restart without executing a rewrite step, and so it rejects according to our definition (see the second paragraph of Sect. 3). Finally, if after executing a rewrite step another factor of the form $aca$ or $bcb$ is encountered, then $M_r$ executes another rewrite step in the same cycle, and thus, it rejects the input. In fact, if $|w|_c > 1$, then $M_r$ will not be able to execute a single successful cycle. It now follows easily that $L(M_r) = L_r$ holds.

On the other hand, the complement $\bar{L}_r$ of $L_r$ is not accepted by any stl-det-RRW-automaton. Assume that $M = (\Sigma, \Sigma, \mathrm{¢}, \$, k, \delta)$ is a stl-det-RRW-automaton that accepts the complement $\bar{L}_r$ of $L_r$, and let $w \in \{a,b\}^*$ be some word that contains all words from $\{a,b\}^k$ as factors. Assume first that $\delta(\mathrm{¢}\cdot u) = \mathrm{¢}\cdot v$ for some words $u$ and $v$. Then we consider the accepting computation of $M$ on input $uwwcw^Ru^R$. Clearly, this cannot just be a tail computation. Hence, it begins with the cycle $uwwcw^Ru^R \vdash_M^c vwwcw^Ru^R$, where $M$ restarts either while reading the first $w$, or with $c$ in its window, or on encountering the right delimiter \$, as $w$ contains all words of length $k$ over $\{a,b\}$ as infixes. Therefore, $M$ also executes the cycle $uwcw^Ru^R \vdash_M^c vwcw^Ru^R \in \bar{L}_r$. This, however, contradicts our assumption that $L(M) = \bar{L}_r$, as $uwcw^Ru^R \in L_r$. Next assume that $\delta(u) = v$ for some words $u$ and $v$. Since $M$ cannot rewrite at the left end of the tape, we obtain a contradiction for the input $uu \in \bar{L}_r$, on which $M$ will execute two rewrite steps, thus rejecting it. Finally, assume that the only rewrite transitions of $M$ are of the form $\delta(u\cdot\$) = v\cdot\$$ for some words $u$ and $v$. Since $M$ cannot rewrite without reading the right endmarker, we obtain a contradiction because of the cycle $u^Rwcw^Ru \vdash_M^c u^Rwcw^Rv$, since $u^Rwcw^Rv \in \bar{L}_r$, while $u^Rwcw^Ru \in L_r$. It follows that $\bar{L}_r$ is indeed not accepted by any stl-det-RRW-automaton. □

From Lemmas 5, 7, and 8 we see that the families $\mathscr{L}$(stl-det-2-RR(W)) would not be closed under complementation, either, if stateless two-phase RRW-automata were required to execute accept steps only at the right of the tape.

For the stateless deterministic R(W)- and 2-RR(W)-automata non-closure under intersection is now immediate. However, we even have the following stronger result.

**Theorem 7** *The families $\mathscr{L}$(stl-det-R(R)(W)) and $\mathscr{L}$(stl-det-2-RR(W)) are not closed under intersection with regular languages.*

*Proof* According to [9] the language

$$L_{\mathrm{expo}} = \left\{ a^{i_0}ba^{i_1}b\cdots a^{i_{n-1}}ba^{i_n} \mid n \geq 0,\ i_0,\ldots,i_n \geq 0,\ \text{and } \exists m \geq 0 : \sum_{j=0}^{n} 2^j \cdot i_j = 2^m \right\} \cup b^*$$

is accepted by a stl-det-RW-automaton, while $L_{\text{expo}} \cap a^* = \{ a^{2^m} \mid m \geq 0 \}$ is not accepted by any RRW-automaton. Analogously, the language $L_{\text{expo}}^{(\phi)} = \phi(L_{\text{expo}})$ is accepted by a stl-det-R-automaton, while $L_{\text{expo}}^{(\phi)} \cap (ab)^* = \{ (ab)^{2^n} \mid n \geq 0 \}$ is not accepted by any RRW-automaton, where $\phi$ is the encoding from Definition 2.

The language $L_{\text{expo}} \cap a^*$ is Church-Rosser, and hence, it is accepted by some stl-det-RRWW-automaton $M = (\{a\}, \Gamma, ¢, \$, k, \delta)$. Now let $M_1$ be the stl-det-RRW-automaton that is obtained from $M$ by declaring all symbols of $\Gamma$ to be input symbols. Then $L(M_1) \in \mathscr{L}$(stl-det-RRW), but $L(M_1) \cap a^* = \{ a^{2^m} \mid m \geq 0 \} \notin \mathscr{L}$(RRW).

In [14] an encoding $\varphi : \Gamma^* \to \{a, b, c, d\}^*$ is presented that depends on $M_1$ such that the image $\varphi(L(M_1))$ is accepted by a deterministic RR-automaton $M_2$. In fact, $M_2$ simulates each step of $M_1$ by a sequence of steps that mirror each action of $M_1$ on the encoded tape contents. Using this encoding it is easily seen that $M_2$ is stateless, if $M_1$ is. Thus, $\varphi(L(M_1)) \in \mathscr{L}$(stl-det-RR). However, $\varphi(L(M_1)) \cap \varphi(a^*) = \{ \varphi(a^{2^n}) \mid n \geq 0 \}$, which is not accepted by any RR-automaton. This completes the proof of Theorem 7. □

Since all language families considered include the regular languages, this proves in particular that none of them is closed under intersection.

### 5.2 Reversal

Next we consider the closure under reversal.

**Theorem 8** *None of the families $\mathscr{L}$(stl-det-R(R)(W)) or $\mathscr{L}$(stl-det-2-RR(W)) is closed under reversal.*

*Proof* First we consider the case of stateless deterministic R(W)-automata. A stl-det-R-automaton $M$ for the language $L_{acab} = \{ a^m c^n a^{n+r} b^{m+r} \mid m, n \geq 1, r \geq 0 \}$ proceeds as follows. In a first phase it performs cycles in which the window is moved across the leading $a$'s and $c$'s until it sees the factor $ccaa$, from which it deletes the factor $ca$. The first phase is completed when there is only one $c$ left. Now the remaining input is of the form $a^m caa^r b^{m+r}$. In a second phase, $M$ performs cycles in which the window is moved across the leading $a$'s followed by $caa^r$ until it sees the factor $aabb$, from which it deletes the factor $ab$. The second phase is completed when there is only one $a$ in between the $c$ and the $b$'s. Now the remaining input is of the form $a^m cab^m$. In a final phase, $M$ performs cycles in which the window is moved across the leading $a$'s until it sees two $a$'s followed by $cabb$. It now deletes the second $a$ and the first $b$. This phase is completed when the input is reduced to $acab$, which is accepted. $M$ can distinguish between these phases based on the contents of its window.

Next, assume that $M_R = (\Sigma, \Sigma, ¢, \$, k, \delta)$ is a stl-det-RW-automaton accepting the reversal of $L_{acab}$. Given an input $b^{m+r} a^{n+r} c^n a^m$ with large $m, n, r$, $M_R$ cannot accept in a tail computation. Furthermore, it cannot rewrite just one type of symbols without violating the correctness preserving property. Now assume that $\delta(b^i a^j) = b^{i-l} a^{j-l}$, where $i + j = k$ and $l \geq 1$. Then after at most $r + 1$ cycles the correctness preserving property is violated, as a word of the form $b^{m-s} a^{n-s} c^n a^m$ is obtained for some $s \geq 1$. Therefore, $M_R$ cannot rewrite in this situation, which means that $M_R$ can never delete one of the leading $b$'s and, thus, cannot check the correct suffix length $m$. We obtain a contradiction, showing that the reversal of the language $L_{acab}$ is not accepted by any stl-det-RW-automaton.

Now we turn to the case of stateless deterministic RR(W)-automata. In the proof of Theorem 6 it is shown that the language $L_r = \left\{ wcw^R \mid w \in \{a, b\}^* \right\}$ is accepted by a stl-det-RR-automaton. Here we consider the language $\hat{L}_r = L_r \cup (L_r \cdot d \cdot \{a, b, c, d\}^*)$,

that is, the union of $L_r$ and the marked concatenation of $L_r$ with $\{a, b, c, d\}^*$. The following extension of the construction from the proof of Theorem 6 gives a stl-det-RR-automaton for this language. Here $x \in \{a, b, c\}$, $y, z \in \{a, b, \varepsilon\}$, and $p \in \{aad, abd, bad, bbd, d \cdot \$\} \cup ad \cdot \{a, b, c, d, \$\} \cup bd \cdot \{a, b, c, d, \$\} \cup d \cdot \{a, b, c, d\} \cdot \{a, b, c, d, \$\}$:

$$
\begin{array}{llll}
(1)\ \delta(\mathfrak{c} \cdot c \cdot \$) = \mathsf{Accept}, & (7)\ \delta(aax) & = \mathsf{MVR}, \\
(2)\ \delta(\mathfrak{c} \cdot cd) = \mathsf{Accept}, & (8)\ \delta(abx) & = \mathsf{MVR}, \\
(3)\ \delta(\mathfrak{c} \cdot ax) = \mathsf{MVR}, & (9)\ \delta(bax) & = \mathsf{MVR}, \\
(4)\ \delta(\mathfrak{c} \cdot bx) = \mathsf{MVR}, & (10)\ \delta(bbx) & = \mathsf{MVR}, \\
(5)\ \delta(aca) = c, & (11)\ \delta(yz \cdot \$) & = \mathsf{Restart}, \\
(6)\ \delta(bcb) = c, & (12)\ \delta(p) & = \mathsf{Restart}.
\end{array}
$$

Now assume that $M$ is a stl-det-RRW-automaton for $\hat{L}_r^R$ with window size $k$, and let $w \in \{a, b\}^*$ be a word that contains all words from $\{a, b\}^k$ as factors. On input $wcw^R$, $M$ executes an accepting computation, which cannot just consist of an accepting tail. Thus, it begins with a cycle of the form $wcw^R \vdash_M^c z$. From the correctness preserving property for $M$ we see that $z = w_1 w_3 c w_3^R w_1^R$, where $w = w_1 w_2$ and $w_3 c w_3^R$ is obtained from $w_2 c w_2^R$ by the rewrite step executed in the above cycle. Therefore, on input $wcw^R wcw^R dwcw^R \in \hat{L}_r^R$, $M$ will execute (at least) two rewrite operations in the first cycle, therewith rejecting this input. This contradicts our assumption that $L(M) = \hat{L}_r^R$ holds.

Finally we give the proof for stl-det-2-RR(W)-automata. By Lemma 5 the language $L_{\text{expo}}^{(1)}$ is accepted by a stl-det-2-RRW-automaton. However, its reversal

$$
\left(L_{\text{expo}}^{(1)}\right)^R = \left\{ a^{2^n} \mid n \geq 0 \right\} \cup \left\{ a^i b a^j \mid i, j \geq 0, \text{ and } \exists m \geq 0 : 2 \cdot i + j = 2^m \right\}
$$

cannot even be accepted by a deterministic RRW-automaton with states. Assume that $M = (Q, \{a, b\}, \{a, b\}, \mathfrak{c}, \$, q_0, k, \delta)$ is a det-RRW-automaton such that $(L_{\text{expo}}^{(1)})^R = L(M)$ holds. Consider the word $w = a^{2^n} \in (L_{\text{expo}}^{(1)})^R$, where $n > 0$ is sufficiently large. As $M$ cannot accept $w$ in a tail computation, the accepting computation of $M$ on input $w$ starts with a cycle $w \vdash_M^c w_1$. Then $w_1 \in (L_{\text{expo}}^{(1)})^R$, and it follows from the fact that each rewrite step of $M$ is length-reducing that $w_1 = a^i b a^{2^n - 2i}$ for some $i$ satisfying $2 \leq i < k/2$. Hence, $M$ transforms a prefix $a^m$ of $w$ into the word $a^i b a^{m-2i}$, where $m \leq 2k$. Now consider the input $z = a^{2^n} b \in (L_{\text{expo}}^{(1)})^R$. Again $M$ cannot accept $z$ in a tail computation. Thus, the accepting computation of $M$ on input $z$ begins with a cycle $z \vdash_M^c z_1$. As $M$ is deterministic, this computation begins with the transformation above, which implies that $z_1 = a^i b a^{2^n - 2i} b$. As $z_1 \notin (L_{\text{expo}}^{(1)})^R$, this contradicts our assumption that $(L_{\text{expo}}^{(1)})^R = L(M)$ holds. It follows that $(L_{\text{expo}}^{(1)})^R$ is not accepted by any deterministic RRW-automaton.

By Lemma 7, $L_\phi^{(1)}$ is accepted by a stl-det-2-RR-automaton, but by using essentially the same reasoning as above it can be shown that $(L_\phi^{(1)})^R$ is not accepted by any det-RR-automaton with states. □

From the proof above we see that not even the language families $\mathscr{L}$(det-RR(W)) are closed under reversal.

### 5.3 Morphisms

This subsection is devoted to disprove the closure of all language families under consideration with respect to morphisms, even non-erasing morphisms, and with respect to inverse morphisms.

**Theorem 9** *None of the families $\mathscr{L}$(stl-det-R(R)(W)) or $\mathscr{L}$(stl-det-2-RR(W)) is closed under non-erasing morphisms.*

*Proof* We consider the language $L_{dm} = \{ ca^n\bar{c}b^n \mid n \geq 0 \} \cup \{ da^n\bar{d}b^{2n} \mid n \geq 0 \}$, which is accepted by all types of automata in question. Exemplarily, we present a stl-det-RR-automaton $M$ for this language:

$$
\begin{array}{llll}
(1) & \delta(\text{¢} \cdot c\bar{c} \cdot \$) = \mathsf{Accept}, & (13) & \delta(aa\bar{d}b) = \mathsf{MVR}, \\
(2) & \delta(\text{¢} \cdot d\bar{d} \cdot \$) = \mathsf{Accept}, & (14) & \delta(da\bar{d}b) = \mathsf{MVR}, \\
(3) & \delta(\text{¢} \cdot caa) = \mathsf{MVR}, & (15) & \delta(aaaa) = \mathsf{MVR}, \\
(4) & \delta(\text{¢} \cdot daa) = \mathsf{MVR}, & (16) & \delta(aa\bar{c}b) = a\bar{c}, \\
(5) & \delta(\text{¢} \cdot ca\bar{c}) = \mathsf{MVR}, & (17) & \delta(ca\bar{c}b) = c\bar{c}, \\
(6) & \delta(\text{¢} \cdot da\bar{d}) = \mathsf{MVR}, & (18) & \delta(a\bar{d}bb) = \bar{d}, \\
(7) & \delta(caaa) = \mathsf{MVR}, & (19) & \delta(bbbb) = \mathsf{Restart}, \\
(8) & \delta(daaa) = \mathsf{MVR}, & (20) & \delta(bbb \cdot \$) = \mathsf{Restart}, \\
(9) & \delta(caa\bar{c}) = \mathsf{MVR}, & (21) & \delta(bb \cdot \$) = \mathsf{Restart}, \\
(10) & \delta(daa\bar{d}) = \mathsf{MVR}, & (22) & \delta(b \cdot \$) = \mathsf{Restart}, \\
(11) & \delta(aaa\bar{c}) = \mathsf{MVR}, & (23) & \delta(\$) = \mathsf{Restart}. \\
(12) & \delta(aaa\bar{d}) = \mathsf{MVR}, & &
\end{array}
$$

It is easily verified that $L(M) = L_{dm}$ holds. Now let $h$ be the non-erasing morphism that is defined by taking $h(a) = a$, $h(b) = b$, $h(c) = c$, $h(d) = d$, $h(\bar{c}) = ab$, and $h(\bar{d}) = abb$. Then $h(L_{dm}) = \{ ca^nb^n \mid n \geq 1 \} \cup \{ da^nb^{2n} \mid n \geq 1 \}$, which equals the language $L_d$ except for the case $n = 0$ (see Sect. 4). As in the proof of Lemma 4 it can be shown that this language is not accepted by any stateless 2-RRW-automaton, either, and therewith not by any automaton of any of the types considered here. □

The non-closure of $\mathscr{L}$(stl-det-R) under inverse morphisms has actually already been shown in Lemma 20 (d) and (e) of [11]. In fact, the language $L_{\text{expo}}$ is not accepted by any RR-automaton, while its morphic image $L_{\text{expo}}^{(\phi)} = \phi(L_{\text{expo}})$ even belongs to $\mathscr{L}$(stl-det-R), where $\phi : \{a, b\}^* \to \{a, b\}^*$ is the morphism from Definition 2. Since this morphism is injective, we obtain the following corollary.

**Corollary 5** *The families $\mathscr{L}$(stl-det-R) and $\mathscr{L}$(stl-det-2-RR) are not closed under inverse morphisms.*

In fact, none of the language classes studied here is closed under inverse morphisms. First, we extend this result to stateless deterministic RW-automata and 2-RRW-automata.

**Lemma 9** *The families $\mathscr{L}$(stl-det-RW) and $\mathscr{L}$(stl-det-2-RRW) are not closed under inverse morphisms.*

*Proof* We utilize the language

$$L_e = \{ c(ac)^m(aee)^{n-m}b^n \mid 0 \leq m \leq n \} \cup \{ d(ad)^m(aee)^{n-m}b^{2n} \mid 0 \leq m \leq n \},$$

which belongs to $\mathscr{L}$(stl-det-RW) by Lemma 20 (a) of [11]. Let $h$ be the morphism that is defined by $h(a) = aee$, $h(b) = b$, $h(c) = c$, $h(d) = d$, and $h(e) = a$. Then $h$ is injective, and it is easily seen that

$$h^{-1}(L_e) = \{ c(ec)^m a^{n-m}b^n \mid 0 \leq m \leq n \} \cup \{ d(ed)^m a^{n-m}b^{2n} \mid 0 \leq m \leq n \}.$$

Now assume that $h^{-1}(L_e)$ is accepted by a stl-det-RW-automaton $M$. Then we observe that the inputs of the form $ca^nb^n$ or $da^nb^{2n}$ (that is, those words for which $m = 0$ holds) cannot be rewritten into words of the form $c(ec)^m a^{n-m}b^n$ or $d(ed)^m a^{n-m}b^{2n}$. This follows from the fact that the corresponding rewrite would have to be applied to the prefix $ca^n$ of $ca^nb^n$ or $da^n$ of $da^nb^{2n}$, leaving the suffix $b^n$ or $b^{2n}$ untouched. However, for all $m \geq 1$, $c(ec)^m a^{n-m}b^n$ or $d(ed)^m a^{n-m}b^{2n}$ is strictly longer than $ca^nb^n$ or $da^nb^{2n}$, respectively. Hence, automaton $M$ rewrites $ca^nb^n$ into $ca^{n-i}b^{n-i}$ and $da^nb^{2n}$ into $da^{n-j}b^{2n-2j}$ for some $i, j \geq 1$. Now modify $M$ into a stl-det-RW-automaton $M'$ by requiring that $M'$ halts without accepting as soon as it encounters an occurrence of the symbol $e$. Then $M'$ accepts the language $L(M) \cap \{a, b, c, d\}^* = h^{-1}(L_e) \cap \{a, b, c, d\}^* = \{\, ca^nb^n \mid n \geq 0 \,\} \cup \{\, da^nb^{2n} \mid n \geq 0 \,\} = L_d$. However, by Lemma 4, $L_d$ is not even accepted by any stateless 2-RRW-automaton. □

The next lemma concludes the investigation of closures under inverse morphisms.

**Lemma 10** *The families $\mathscr{L}$(stl-det-RR(W)) are not closed under inverse morphisms.*

*Proof* The language

$$L_{abam} = \left\{\, a^m(bc)^{m+n}a^n \mid m, n \geq 0 \,\right\} \cup \left\{\, b^m(bc)^{n-m}a^n \mid 0 \leq m \leq n \,\right\}$$

is accepted by the following stl-det-RR-automaton $M$:

| | | | | | |
|---|---|---|---|---|---|
| (1) | $\delta(¢ \cdot \$)$ | = Accept, | (14) | $\delta(abcb)$ | $= b$, |
| (2) | $\delta(¢ \cdot ba \cdot \$)$ | = Accept, | (15) | $\delta(abc \cdot \$)$ | $= \$$, |
| (3) | $\delta(¢ \cdot aaa)$ | = MVR, | (16) | $\delta(¢ \cdot bcb)$ | $= ¢ \cdot bb$, |
| (4) | $\delta(¢ \cdot aab)$ | = MVR, | (17) | $\delta(¢ \cdot bca)$ | $= ¢ \cdot ba$, |
| (5) | $\delta(¢ \cdot abc)$ | = MVR, | (18) | $\delta(¢ \cdot bba)$ | $= ¢ \cdot b$, |
| (6) | $\delta(¢ \cdot bbb)$ | = MVR, | (19) | $\delta(bbcb)$ | $= bbb$, |
| (7) | $\delta(¢ \cdot bbc)$ | = MVR, | (20) | $\delta(bbca)$ | $= bba$, |
| (8) | $\delta(aaaa)$ | = MVR, | (21) | $\delta(bbaa)$ | $= ba$, |
| (9) | $\delta(aaab)$ | = MVR, | (22) | $\delta(cbcb)$ | = Restart, |
| (10) | $\delta(aabc)$ | = MVR, | (23) | $\delta(cbca)$ | = Restart, |
| (11) | $\delta(bbbb)$ | = MVR, | (24) | $\delta(caaa)$ | = Restart, |
| (12) | $\delta(bbbc)$ | = MVR, | (25) | $\delta(cbc \cdot \$)$ | = Restart, |
| (13) | $\delta(bbba)$ | = MVR, | (26) | $\delta(ca^i \cdot \$)$ | = Restart for all $0 \leq i \leq 2$, |
| | | | (27) | $\delta(a^i \cdot \$)$ | = Restart for all $0 \leq i \leq 3$. |

Given an input $w \in \{a, b, c\}^*$ as input, $M$ proceeds as follows. If $w = \varepsilon$ or $w = ba$ $(= b^1(bc)^0a^1)$, then $M$ accepts immediately by (1) or (2), and if $w = bcbw_2$ or $w = bcaw_2$, then $M$ executes a rewrite step that deletes the first occurrence of the symbol $c$ by (16) or (17). It is, however, easily seen that $w = bcbw_2$ $(w = bcaw_2)$ belongs to the language $L_{abam}$ if and only if the resulting word $bbw_2$ $(baw_2)$ belongs to $L_{abam}$. In fact, if $w = bcbw_2 \in L_{abam}$, then $w_2 = c(bc)^n a^{n+2}$, and if $w = bcaw_2 \in L_{abam}$, then $w_2 = \varepsilon$. In all resulting cases $M$ performs a restart in the next step. Finally, if $w = w_1w_2$ for some prefix $w_1 \in \{a^3, a^2b, abc, b^3, b^2c\}$, then $M$ performs a move-right step using (3) to (7). In fact, if $w_1$ starts with an $a$, then $M$ moves right across all leading $a$'s until it detects the factor $abcb$ or $abc \cdot \$$, from which it then deletes the prefix $abc$ by (14) or (15). If $w \in L_{abam}$, then so is the resulting word, and in addition, in all possible cases $M$ performs a restart in the next step. If no such factor is found, then $M$ either gets stuck (for example, this happens should it encounter the factor $aaac$), or it reaches the right sentinal \$ and executes a restart, without

having performed a rewrite step first. In both these cases $M$ is said to reject the input $w$. By cycling through these steps, $M$ deletes a prefix of the form $a^m(bc)^m$ from $w$.

On the other hand, if $w_1$ starts with a $b$, then $M$ moves right across all leading $b$'s until it detects the factor $bbcb$, $bbca$, or $bbaa$. In the first two cases it then deletes the letter $c$ by (19) or (20), while in the latter case it deletes the factor $ab$ using (21). Again, if $w \in L_{abam}$, then so is the resulting word, and in the next step $M$ performs a restart. By cycling though these steps, $M$ transforms a word of the form $b^m(bc)^s a^n$ first into the word $b^{m+s}a^n$, and then it transforms $b^{m+s}a^n$ into $ba$, if $m+s=n$ holds. Thus, it follows that $L(M) = L_{abam}$ holds as claimed.

Now let $h$ be the morphism that is defined by $h(a) = a$ and $h(b) = bc$. Then $h^{-1}(L_{abam}) = \{ a^m b^{m+n} a^n \mid m, n \geq 0 \} = L_{aba}$, which is not accepted by any stateless deterministic RRW-automaton by Lemma 1. Thus, the families $\mathscr{L}$(stl-det-RR) and $\mathscr{L}$(stl-det-RRW) are not closed under inverse morphisms. □

### 5.4 Catenation operations

We conclude this section by showing that none of the language families considered here is closed under concatenation or iteration.

**Lemma 11** *The families $\mathscr{L}$(stl-det-R(W)) and $\mathscr{L}$(stl-det-2-RR(W)) are neither closed under concatenation nor under iteration.*

*Proof* Consider the language

$$L_f = \left\{ a^n b^n \mid n \geq 0 \right\} \cup a^* \cup \left\{ b^n c^n \mid n \geq 0 \right\} \cup c^*,$$

which is accepted by the stateless deterministic R-automaton, which is obtained by generalizing the construction from Example 1 as follows:

$$\begin{array}{llll}
(1) & \delta(¢ \cdot \$) = \mathsf{Accept}, & (12) & \delta(¢ \cdot ccc) = \mathsf{MVR}, \\
(2) & \delta(¢ \cdot ab \cdot \$) = \mathsf{Accept}, & (13) & \delta(aaaa) = \mathsf{MVR}, \\
(3) & \delta(¢ \cdot a \cdot \$) = \mathsf{Accept}, & (14) & \delta(aaab) = \mathsf{MVR}, \\
(4) & \delta(¢ \cdot aa \cdot \$) = \mathsf{Accept}, & (15) & \delta(bbbb) = \mathsf{MVR}, \\
(5) & \delta(¢ \cdot bc \cdot \$) = \mathsf{Accept}, & (16) & \delta(bbbc) = \mathsf{MVR}, \\
(6) & \delta(¢ \cdot c \cdot \$) = \mathsf{Accept}, & (17) & \delta(cccc) = \mathsf{MVR}, \\
(7) & \delta(¢ \cdot cc \cdot \$) = \mathsf{Accept}, & (18) & \delta(aabb) = ab, \\
(8) & \delta(¢ \cdot aaa) = \mathsf{MVR}, & (19) & \delta(aaa \cdot \$) = aa \cdot \$, \\
(9) & \delta(¢ \cdot aab) = \mathsf{MVR}, & (20) & \delta(bbcc) = bc, \\
(10) & \delta(¢ \cdot bbb) = \mathsf{MVR}, & (21) & \delta(ccc \cdot \$) = cc \cdot \$. \\
(11) & \delta(¢ \cdot bbc) = \mathsf{MVR}, & &
\end{array}$$

Now we consider the concatenation $L_f \cdot L_f$. Assume that $M'$ is a stateless determininistic 2-RRW-automaton for this language. An input of the form $a^n b^n c^n$ with large $n$ cannot be accepted by $M'$ in a tail computation, that is, it has to be rewritten. Because of the correctness preserving property, the word obtained by this rewrite operation must belong to $L_f \cdot L_f$. Thus, either a factor of the form $a^i b^i$ or a factor of the form $b^i c^i$ for some small integer $i > 0$ is deleted. But due to the deterministic behavior of $M'$ any such rewrite violates the correctness preserving property either on input $a^{n+1}b^n c^n$ or on input $a^n b^n c^{n+1}$. We therefore conclude that $L_f \cdot L_f$ is not accepted by any stateless deterministic 2-RRW-automaton. Hence, the families $\mathscr{L}$(stl-det-R(W)) and $\mathscr{L}$(stl-det-2-RR(W)) are not closed under concatenation. Since $L_f \cdot L_f$ is also a subset of the iteration $(L_f)^*$, these families are not closed under iteration, either. □

**Table 1** Closure properties of language families specified by stateless deterministic restarting automata

| | Operation | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\cup$ | $\cap$ | $\sim$ | $\cap_{reg}$ | $R$ | $\cdot$ | $*$ | $h^{-1}$ | $h_\varepsilon$ |
| $\mathscr{L}$(stl-det-R) | − | − | + | − | − | − | − | − | − |
| $\mathscr{L}$(stl-det-RW) | − | − | + | − | − | − | − | − | − |
| $\mathscr{L}$(stl-det-RR) | − | − | − | − | − | − | − | − | − |
| $\mathscr{L}$(stl-det-RRW) | − | − | − | − | − | − | − | − | − |
| $\mathscr{L}$(stl-det-2-RR) | − | − | + | − | − | − | − | − | − |
| $\mathscr{L}$(stl-det-2-RRW) | − | − | + | − | − | − | − | − | − |

+ The language family is closed under the operation under consideration, − that it is *not* closed

Basically, the reasoning of the next lemma has been used before.

**Lemma 12** *The families* $\mathscr{L}$(stl-det-*RR(W)*) *are neither closed under concatenation nor under iteration.*

*Proof* In the proof of Theorem 6 it is shown that the language $L_r = \{ wcw^R \mid w \in \{a, b\}^* \}$ is accepted by some stl-det-RR-automaton.

Now assume that the concatenation $L_r \cdot L_r$ is accepted by a stl-det-RRW-automaton $M$ with window size $k$, and let $x = wcw^R wcw^R \in L_r \cdot L_r$ be an input, where $w \in \{a, b\}^*$ is chosen in such a way that it contains all words from $\{a, b\}^k$ as factors. Certainly $M$ cannot accept input $x$ in a tail computation, that is, its accepting computation on input $x$ begins with a cycle $x \vdash_M^c y$. Within this cycle a rewrite operation can only be performed with an occurrence of the symbol $c$ inside the window, as a rewrite at any other position would violate the correctness preserving property for $M$. By the choice of $w$ this implies that $M$ actually performs two rewrite operations in the first cycle. Hence, it rejects on input $x$, that is, $L(M) \neq L_r \cdot L_r$. Thus, the families $\mathscr{L}$(stl-det-RR(W)) are not closed under concatenation. Since $L_r \cdot L_r$ is also a subset of the iteration $(L_r)^*$, the argument above also shows that these families are not closed under iteration, either. □

## 6 Concluding remarks

We have studied various different types of stateless deterministic restarting automata. While for RWW-automata the restriction to stateless variants is straightforward, we have seen that for RRWW-automata there are different ways of realizing this restriction. To us the two-phase variants are more appealing, but the other variants remain interesting, as their relationship to the corresponding stateless RW-automata is still unsolved. For all types of restarting automata considered, auxiliary symbols can easily compensate for the loss of states, but without them the stateless deterministic variants are strictly weaker than those with states. In particular, for all types of stateless deterministic restarting automata without auxiliary symbols considered, the corresponding language families form anti-AFLs. Table 1 summarizes the (non-) closure properties that we obtained for these families of languages. It is expected that many of these results also carry over to the nondeterministic case.

## References

1. Buntrock, G., Otto, F.: Growing context-sensitive languages and Church-Rosser languages. Inform. Comput. **141**, 1–36 (1998)

2. Culy, C.: Formal properties of natural language and linguistic theories. Linguist. Philos. **19**, 599–617 (1996)
3. Frisco, P., Ibarra, O.H.: On stateless multihead finite automata and multihead pushdown automata. In: Diekert, V., Nowotka, D. (eds.) DLT 2009, Proceedings, Lecture Notes in Computer Science, vol. 5583, pp. 240–251. Springer, Berlin (2009)
4. Holzer, M., Kutrib, M., Reimann, J.: Non-recursive trade-offs for deterministic restarting automata. J. Autom. Lang. Comb. **12**, 195–213 (2007)
5. Harrison, M.: Introduction to Formal Language Theory. Addison-Wesley, Reading, MA (1978)
6. Ibarra, O.H., Dang, Z., Egecioglu, Ö: Catalytic P systems, semilinear sets, and vector addition systems. Theor. Comput. Sci. **312**, 379–399 (2004)
7. Ibarra, O.H., Karhumäki, J., Okhotin, A.: On stateless multihead automata: Hierarchies and the emptiness problem. In: Laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L. (eds.) LATIN 2008, Proceedings, Lecture Notes in Computer Science, vol. 4957, pp. 94–105. Springer, Berlin (2008)
8. Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata. In: Reichel, H. (ed.) FCT 1995, Proceedings, Lecture Notes in Computer Science, vol. 965, pp. 283–292. Springer, Berlin (1995)
9. Kutrib, M., Messerschmidt, H., Otto, F.: On stateless two-pushdown automata and restarting automata. In: Csuhaj-Varjú, E., Ésik, Z. (eds.) Automata and Formal Languages, AFL 2008, Proceedings, pp. 257–268. Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest (2008)
10. Kutrib, M., Messerschmidt, H., Otto, F.: On stateless deterministic restarting automata. In: Nielsen, M., Kučera, A., Miltersen, P.B., Palamidessi, C., Tuma, P., Valencia, F. (eds.) SOFSEM 2009: Theory and Practice of Computer Science, Proceedings, Lecture Notes in Computer Science, vol. 5404, pp. 353–364. Springer, Berlin (2009)
11. Kutrib, M., Messerschmidt, H. Otto, F.: On stateless two-pushdown automata and restarting automata. Extended version of Kutrib et al. (2008). (Submitted for publication)
12. Kutrib, M., Reimann, J.: Succinct description of regular languages by weak restarting automata. Inform. Comput. **206**, 1152–1160 (2008)
13. Niemann, G., Otto, F.: The Church-Rosser languages are the deterministic variants of the growing context-sensitive languages. Inform. Comput. **197**, 1–21 (2005)
14. Otto, F.: Restarting automata. In: Ésik, Z., Martin-Vide, C., Mitrana, V. (eds.) Recent Advances in Formal Languages and Applications, Studies in Computational Intelligence, vol. 25, pp. 269–303. Springer, Berlin (2006)
15. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. **3**, 114–125 (1959)
16. Yang, L., Dang, Z., Ibarra, O.H.: On stateless automata and P systems. In: Workshop on Automata for Cellular and Molecular Computing, MTA SZTAKI, pp. 144–157 (2007)