On Some Decision Problems for Stateless Deterministic Ordered Restarting Automata

Kent Kwee and Friedrich Otto(✉)

Fachbereich Elektrotechnik/Informatik, Universität Kassel, 34109 Kassel, Germany
{kwee,otto}@theory.informatik.uni-kassel.de

Abstract. The stateless deterministic ordered restarting automata accept exactly the regular languages, and it is known that the trade-off for turning a stateless deterministic ordered restarting automaton into an equivalent DFA is at least double exponential. Here we show that the trade-off for turning a stateless deterministic ordered restarting automaton into an equivalent unambiguous NFA is exponential, which yields an upper bound of $2^{2^{O(n)}}$ for the conversion into an equivalent DFA, thus meeting the lower bound up to a constant. Based on the new transformation we then show that many decision problems, such as emptiness, finiteness, inclusion, and equivalence, are PSPACE-complete for stateless deterministic ordered restarting automata.

Keywords: Restarting automaton · Ordered rewriting · Descriptional complexity · Decision problem

1 Introduction

The *deterministic ordered restarting automaton* (or *det-ORWW-automaton*) was introduced in [9] in the setting of picture languages. While the nondeterministic variant of this type of automaton even accepts some languages that are not context-free, it has been shown in [9] that the deterministic variant accepts exactly the regular languages.

In [10] an investigation of the descriptional complexity of the det-ORWW-automaton was initiated. It was shown that each det-ORWW-automaton can be simulated by an automaton of the same type that has only a single state, which means that for these automata, states are actually not needed. Accordingly, such an automaton is called a *stateless* det-ORWW-automaton (stl-det-ORWW-automaton). For these automata, the size of their working alphabets can be taken as a measure for their descriptional complexity, and it has been shown that these automata are polynomially related in size to the weight-reducing Hennie machines studied by Průša in [12]. Actually, for $n \geq 1$, there exists a regular language that is accepted by a stl-det-ORWW-automaton of size $O(n)$ such that each DFA for this language has size at least 2^{2^n}. On the other hand, each stl-det-ORWW-automaton of size n can be simulated by a DFA of size $2^{2^{O(n^2 \cdot \log n)}}$. Thus, there is a huge gap between the upper and lower bounds.

J. Shallit and A. Okhotin (Eds.): DCFS 2015, LNCS 9118, pp. 165–176, 2015.
DOI: 10.1007/978-3-319-19225-3_14

Here we present a new construction that, for a stl-det-ORWW-automaton of size n, yields an equivalent unambiguous NFA of size $2^{O(n)}$, which implies that there is an equivalent DFA of size $2^{2^{O(n)}}$. Actually, we will show that these bounds are sharp (up to the O-notation). We then exploit our construction to establish that many basic decision problems, like emptiness, universality, finiteness, inclusion, and equivalence, are PSPACE-complete for stl-det-ORWW-automata. In addition, we consider the problem of deciding, given a stl-det-ORWW-automaton, whether the language accepted belongs to a certain subclass of the regular languages. For the subclasses of strictly locally k-testable languages ($k \geq 1$), nilpotent languages, combinatorial languages, and some others, we obtain that the corresponding decision problems are PSPACE-complete, too.

This paper is structured as follows. In Sect. 2, we introduce the stl-det-ORWW-automata, and we restate the main results on them from [10]. Then, in Sect. 3, we present the announced construction of an NFA from a given stl-det-ORWW-automaton, and in Sect. 4 we consider the decision problems mentioned above. The paper closes with Sect. 5, which summarizes our results briefly and states a number of open problems for future work.

2 Stateless Deterministic Ordered Restarting Automata

A *stateless deterministic ordered restarting automaton* (stl-det-ORWW-automaton) is a one-tape machine that is described by a 6-tuple $M = (\Sigma, \Gamma, \triangleright, \triangleleft, \delta, >)$, where Σ is a finite input alphabet, Γ is a finite tape alphabet such that $\Sigma \subseteq \Gamma$, the symbols $\triangleright, \triangleleft \notin \Gamma$ serve as markers for the left and right border of the work space, respectively,

$$\delta : (((\Gamma \cup \{\triangleright\}) \cdot \Gamma \cdot (\Gamma \cup \{\triangleleft\})) \cup \{\triangleright\triangleleft\}) \dashrightarrow \{\mathsf{MVR}\} \cup \Gamma \cup \{\mathsf{Accept}\}$$

is the (partial) *transition function*, and $>$ is a *partial ordering* on Γ. The transition function describes three different types of transition steps:

(1) A *move-right step* has the form $\delta(a_1a_2a_3) = \mathsf{MVR}$, where $a_1 \in \Gamma \cup \{\triangleright\}$ and $a_2, a_3 \in \Gamma$. It causes M to shift the window one position to the right. Observe that no move-right step is possible, if the window contains the symbol $\triangleleft$.
(2) A *rewrite/restart step* has the form $\delta(a_1a_2a_3) = b$, where $a_1 \in \Gamma \cup \{\triangleright\}$, $a_2, b \in \Gamma$, and $a_3 \in \Gamma \cup \{\triangleleft\}$ such that $a_2 > b$ holds. It causes M to replace the symbol a_2 in the middle of its window by the symbol b and to restart.
(3) An *accept step* has the form $\delta(a_1a_2a_3) = \mathsf{Accept}$, where $a_1 \in \Gamma \cup \{\triangleright\}$, $a_2 \in \Gamma$, and $a_3 \in \Gamma \cup \{\triangleleft\}$. It causes M to halt and accept. In addition, we allow an accept step of the form $\delta(\triangleright\triangleleft) = \mathsf{Accept}$.

If $\delta(u)$ is undefined for some word u, then M necessarily halts, when it sees u in its window, and we say that M *rejects* in this situation. Further, the letters in $\Gamma \smallsetminus \Sigma$ are called *auxiliary symbols*.

A *configuration* of a stl-det-ORWW-automaton M is a pair of words (α, β), where $|\beta| \geq 3$, and either $\alpha = \lambda$ (the empty word) and $\beta \in \{\triangleright\} \cdot \Gamma^+ \cdot \{\triangleleft\}$ or

$\alpha \in \{\rhd\} \cdot \Gamma^*$ and $\beta \in \Gamma \cdot \Gamma^+ \cdot \{\lhd\}$; here $\alpha\beta$ is the current content of the tape, and it is understood that the window contains the first three symbols of β. In addition, we admit the configuration $(\lambda, \rhd\lhd)$. A *restarting configuration* has the form $(\lambda, \rhd w \lhd)$; if $w \in \Sigma^*$, then $(\lambda, \rhd w \lhd)$ is also called an *initial configuration*. Furthermore, we use Accept to denote the *accepting configurations*, which are those configurations that M reaches by an accept step. We let $\vdash_M$ denote the *single-step computation relation* that M induces on the set of configurations, and the *computation relation* $\vdash_M^*$ of M is the reflexive and transitive closure of $\vdash_M$.

Any computation of a stl-det-ORWW-automaton M consists of certain phases. A phase, called a *cycle*, starts in a restarting configuration, the head is moved along the tape by MVR steps until a rewrite/restart step is performed and thus, a new restarting configuration is reached. If no further rewrite operation is performed, any computation necessarily finishes in a halting configuration – such a phase is called a *tail*. By $\vdash_M^c$ we denote the execution of a complete cycle, and $\vdash_M^{c^*}$ is the reflexive transitive closure of this relation. It can be seen as the rewrite relation that M induces on its set of restarting configurations.

An input $w \in \Sigma^*$ is accepted by M if the computation of M which starts with the initial configuration $(\lambda, \rhd w \lhd)$ ends with an accept step. The language consisting of all input words that are accepted by M is denoted by $L(M)$.

As each cycle ends with a rewrite operation, which replaces a symbol a by a symbol b that is strictly smaller than a with respect to the given ordering $>$, we see that each computation of M on an input of length n consists of at most $(|\Gamma| - 1) \cdot n$ cycles and a tail. Thus, M can be simulated by a deterministic single-tape Turing machine in time $O(n^2)$. The following example illustrates the way in which a stl-det-ORWW-automaton works.

Example 1. Let $n \geq 2$ be a fixed integer, and let $M = (\Sigma, \Gamma, \rhd, \lhd, \delta, >)$ be defined by taking $\Sigma = \{a, b\}$ and $\Gamma = \Sigma \cup \{\, a_i, b_i, x_i \mid 1 \leq i \leq n-1 \,\}$, by choosing the ordering $>$ such that $a > a_i > x_j$ and $b > b_i > x_j$ hold for all $1 \leq i, j \leq n-1$, and by defining the transition function δ in such a way that M proceeds as follows: on input $w = w_1 w_2 \cdots w_m$, $w_1, \ldots, w_m \in \Sigma$, M numbers the first $n-1$ letters of w from left to right, by replacing $w_i = a$ (b) by a_i (b_i) for $i = 1, \ldots, n-1$. If $w_n \neq a$, then the computation fails, but if $w_n = a$, then M continues by replacing the last $n-1$ letters of w from right to left using the letters x_1 to x_{n-1}. If the n-th last letter is b or some b_i, then M accepts, otherwise the computation fails again.

Then $L(M) = \{\, w \in \{a, b\}^m \mid m > n,\ w_n = a, \text{ and } w_{m+1-n} = b \,\}$. As shown in [6], every det-RR(1)-automaton for $L(M)$ has at least $O(2^n)$ states. Here a det-RR(1)-automaton is another type of deterministic restarting automaton that characterizes the regular languages (see [7]).

While nondeterministic ORWW-automata are quite expressive, the deterministic variants are fairly weak. Taking the size of the tape alphabet as the measure for the descriptional complexity of a stl-det-ORWW-automaton, the following results are shown in [10].

Theorem 2

(a) For each DFA $A = (Q, \Sigma, q_0, F, \varphi)$, there is a stl-det-ORWW-automaton $M = (\Sigma, \Gamma, \rhd, \lhd, \delta, >)$ such that $L(M) = L(A)$ and $|\Gamma| = |Q| + |\Sigma|$.
(b) For each stl-det-ORWW-automaton M with an alphabet of size n, there exists a DFA A of size $2^{2^{O(n^2 \log n)}}$ such that $L(A) = L(M)$ holds.
(c) For each $n \geq 1$, there exists a regular language $B_n \subseteq \{0, 1, \$\}^*$ such that B_n is accepted by a stl-det-ORWW-automaton over an alphabet of size $O(n)$, but each DFA for accepting B_n has at least 2^{2^n} states.

Thus, there is a double exponential trade-off for converting a stl-det-ORWW-automaton into a DFA. Observe, however, that the gap between the lower and upper bounds is still huge.

3 Simulating a stl-det-ORWW-automaton by an NFA

Here we present our main result, which consists in the construction of an unambiguous NFA A of size $2^{O(n)}$ from a stl-det-ORWW-automaton M of size n such that A accepts the same language as M. In order to simplify this construction, we require that M only accepts on reaching the right sentinel $\lhd$. This is not a restriction, as shown by the following lemma.

Lemma 3. *From a stl-det-ORWW-automaton $M = (\Sigma, \Gamma, \rhd, \lhd, \delta, >)$, one can construct a stl-det-ORWW-automaton $M' = (\Sigma, \Delta, \rhd, \lhd, \delta', >)$ such that $L(M') = L(M)$, $|\Delta| \leq |\Gamma| + 1$, and M' only accepts when its window contains the right sentinel $\lhd$.*

To motivate our main construction we consider an example.

Example 4. Let M be a stl-det-ORWW-automaton on the input alphabet $\Sigma = \{a_1, a_2, a_3, a_4, a_5\}$ and the working alphabet $\Gamma = \Sigma \cup \{b_1, b_2, b_3, b_4, c_1, c_2, c_3, c_4\}$ with the ordering $a_i > b_i > c_i$ for all $1 \leq i \leq 4$, and let the transition function be given by the following table:

$\delta(\rhd a_1 a_2) = b_1,$	$\delta(\rhd b_1 a_2) = \mathsf{MVR},$	$\delta(b_1 a_2 a_3) = \mathsf{MVR},$	$\delta(a_2 a_3 a_4) = b_3,$
$\delta(b_1 a_2 b_3) = b_2,$	$\delta(c_2 c_3 a_4) = \mathsf{MVR},$	$\delta(\rhd c_1 b_2) = \mathsf{MVR},$	$\delta(c_1 b_2 b_3) = \mathsf{MVR},$
$\delta(b_2 b_3 a_4) = c_3,$	$\delta(c_2 c_3 c_4) = \mathsf{MVR},$	$\delta(\rhd c_1 c_2) = \mathsf{MVR},$	$\delta(c_1 c_2 c_3) = \mathsf{MVR},$
$\delta(c_1 b_2 c_3) = c_2,$	$\delta(c_3 a_4 a_5) = b_4,$	$\delta(c_2 c_3 b_4) = \mathsf{MVR},$	$\delta(c_3 b_4 a_5) = c_4,$
$\delta(\rhd b_1 b_2) = c_1,$	$\delta(c_3 c_4 a_5) = \mathsf{MVR},$	$\delta(c_4 a_5 \lhd) = \mathsf{Accept}.$	

Given the word $w = a_1 a_2 a_3 a_4 a_5$ as input, M executes the following accepting computation, where the rewritten letters are underlined:

$$(\lambda, \rhd \underline{a_1} a_2 a_3 a_4 a_5 \lhd) \vdash_M^c (\lambda, \rhd b_1 a_2 \underline{a_3} a_4 a_5 \lhd) \vdash_M^c (\lambda, \rhd b_1 \underline{a_2} b_3 a_4 a_5 \lhd) \vdash_M^c$$
$$(\lambda, \rhd \underline{b_1} b_2 b_3 a_4 a_5 \lhd) \vdash_M^c (\lambda, \rhd c_1 b_2 \underline{b_3} a_4 a_5 \lhd) \vdash_M^c (\lambda, \rhd c_1 \underline{b_2} c_3 a_4 a_5 \lhd) \vdash_M^c$$
$$(\lambda, \rhd c_1 c_2 c_3 \underline{a_4} a_5 \lhd) \vdash_M^c (\lambda, \rhd c_1 c_2 c_3 \underline{b_4} a_5 \lhd) \vdash_M^c (\lambda, \rhd c_1 c_2 c_3 c_4 a_5 \lhd) \vdash_M^* \mathsf{Accept}.$$

To encode this computation in a compact way, we introduce a 3-tuple of vectors $T = (L, W, R)$ for each position on the tape of M, where

– W is a sequence of letters $W = (x_1, x_2, \ldots, x_r)$ over Γ such that $x_1 > x_2 > \cdots > x_r$ using the ordering on Γ defined by M,
– L is a sequence of indices $L = (i_1, \ldots, i_{r-1})$ such that $i_1 \le \cdots \le i_{r-1} \le |\Gamma|$,
– R is a sequence of indices $R = (j_1, \ldots, j_{r-1})$ such that $j_1 \le \cdots \le j_{r-1} \le |\Gamma|$.

The idea is that W encodes the sequence of letters that are produced by M in an accepting computation for a particular field, and L and R encode the information on the neighbouring letters to the left and to the right that are used to perform the corresponding rewrite operations. For the computation above we obtain the following sequence of triples, where Λ denotes an empty sequence:

L_0	W_0	R_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	L_5	W_5	R_5	L_6	W_6	R_6
Λ	$\rhd$	Λ	1	a_1	1	2	a_2	2	1	a_3	1	3	a_4	1	Λ	a_5	Λ	Λ	$\lhd$	Λ
			1	b_1	2	3	b_2	3	2	b_3	1	3	b_4	1						
				c_1			c_2			c_3			c_4							

For example, the triple $(2, b_3, 1) \in (L_3, W_3, R_3)$ means that b_3 is rewritten into c_3, while the left neighbouring field contains the second letter of its sequence W_2, and the right neighbouring field contains the first letter of its sequence W_4.

If a letter is not rewritten at all, like a_5, then the corresponding sequences L and R are empty. In fact, there is a consistency condition that must be met by the sequences R_{i-1} and L_i for each index i, as the rewrites at positions $i-1$ and i are executed in some order, and this order is encoded in these sequences. For example, $L_3 = (1, 2)$, which means that a_3 is rewritten into b_3, while tape field 2 still contains the original letter a_2, and b_3 is rewritten into c_3, while tape field 2 contains the next letter b_2. Thus, before the second rewrite at position 3 can occur, the letter a_2 at position 2 has been rewritten into b_2, which is expressed by the fact that $R_2 = (2, 3)$ starts with the number 2. Finally, the second number in R_2 states that b_2 is rewritten into c_2 only after the second rewrite at position 3 has been performed. Hence, $R_2 = (2, 3)$ and $L_3 = (1, 2)$ lead to the sequence of rewrite steps $(1 : a_3 \to b_3), (2 : a_2 \to b_2), (3 : b_3 \to c_3), (4 : b_2 \to c_2)$. □

To formalize the notion of *compatibility* of two finite non-decreasing sequences of integers $R = (r_1, \ldots, r_k)$ and $L = (\ell_1, \ldots, \ell_s)$, where $k, s \ge 0$, we define a multiset order(R, L) as follows:

$$\mathrm{order}(R, L) = \{\, r_i + i - 1 \mid i = 1, \ldots, k \,\} \cup \{\, \ell_j + j - 1 \mid j = 1, \ldots, s \,\}.$$

Now the pair of sequences (R, L) is called *consistent*, if order$(R, L) = \{1, 2, \ldots, k + s\}$, that is, it is the integer interval $[1, k + s]$. In the example above, we obtain order(R_2, L_3) = order$((2, 3), (1, 2)) = \{2, 4, 1, 3\} = \{1, 2, 3, 4\}$, thus we assign a number between 1 and $4 = |R_2| + |L_3|$ to each of the rewrites at positions $i - 1$ and i, in this way specifying the order in which these rewrites must be executed.

Based on the above ideas, we will now establish the following general result.

Theorem 5. *Let $M = (\Sigma, \Gamma, \rhd, \lhd, \delta_M, >)$ be a stl-det-ORWW-automaton. Then an unambiguous NFA $A = (Q, \Sigma, \Delta_A, q_0, F)$ can be constructed from M such that $L(A) = L(M)$ and $|Q| \in 2^{O(|\Gamma|)}$.*

Proof. Let $M = (\Sigma, \Gamma, \rhd, \lhd, \delta_M, >)$ be a stl-det-ORWW-automaton. At the extra cost of at most one additional tape symbol, we can assume by Lemma 3 that M executes an accept step only when its window contains the right sentinel $\lhd$. Let $n = |\Gamma|$. As a first step we construct an NFA B for the *characteristic language* $L_C(M) = \{ w \in \Gamma^* \mid (\lambda, \rhd w \lhd) \vdash_M^* \mathsf{Accept} \}$ of M, which consists of all words over Γ that M accepts.

The NFA $B = (Q, \Gamma, \Delta_B, q_0, F)$ is constructed as follows:

– The set Q contains the initial state q_0, a designated final state q_F, and all pairs of triples of the form $((L_1, W_1, R_1), (L_2, W_2, R_2))$, where, for $i = 1, 2$,
  - W_i is a sequence of letters $W_i = (w_{i,1}, \ldots, w_{i,k_i})$ from Γ of length $1 \leq k_i \leq n$ such that $w_{i,1} > w_{i,2} > \cdots > w_{i,k_i}$, or $W_i = (\rhd)$ and $k_i = 1$,
  - L_i is a sequence of positive integers $L_i = (l_{i,1}, \ldots, l_{i,k_i-1})$ of length $k_i - 1$ such that $l_{i,1} \leq l_{i,2} \leq \cdots \leq l_{i,k_i-1} \leq n$,
  - R_i is a sequence of positive integers $R_i = (r_{i,1}, \ldots, r_{i,k_i-1})$ of length $k_i - 1$ such that $r_{i,1} \leq r_{i,2} \leq \cdots \leq r_{i,k_i-1} \leq n$,
  - the sequences R_1 and L_2 are consistent, that is, $\text{order}(R_1, L_2) = \{1, 2, \ldots, k_1 + k_2 - 2\}$.

The transition relation Δ_B is given through the following rules, where $x \in \Gamma$ and $((L_{i-1}, W_{i-1}, R_{i-1}), (L_i, W_i, R_i))$, $i = 2, 3$, are states from Q:

– $\Delta_B(q_0, \lambda) \ni q_F$, if $\delta_M(\rhd\lhd) = \mathsf{Accept}$.

– $\Delta_B(q_0, x) \ni ((\Lambda, (\rhd), \Lambda), (L_1, W_1, R_1))$, if $x = w_{1,1}$.

– $\Delta_B\left(((L_1, W_1, R_1), (L_2, W_2, R_2)), x\right) \ni ((L_2, W_2, R_2), (L_3, W_3, R_3))$, if
  1. $x = w_{3,1}$,
  2. $\forall 1 \leq j \leq k_2 - 1 : \delta_M\left(w_{1,l_{2,j}} w_{2,j} w_{3,r_{2,j}}\right) = w_{2,j+1}$,
  3. $\forall 1 \leq j \leq k_3 - 1 : \delta_M\left(w_{1,l_{2,l_{3,j}}} w_{2,l_{3,j}} w_{3,j}\right) = \mathsf{MVR}$, where $l_{2,k_2} = k_1$ is taken, and
  4. $\delta_M\left(w_{1,k_1} w_{2,k_2} w_{3,k_3}\right) = \mathsf{MVR}$.

– $\Delta_B\left(((L_1, W_1, R_1), (L_2, W_2, R_2)), \lambda\right) \ni q_F$, if
  1. R_2 is a sequence of 1's of length $k_2 - 1$,
  2. $\delta_M\left(w_{1,k_1} w_{2,k_2} \lhd\right) = \mathsf{Accept}$, and
  3. $\forall 1 \leq j \leq k_2 - 1 : \delta_M\left(w_{1,l_{2,j}} w_{2,j} \lhd\right) = w_{2,j+1}$.

We will prove that $L(B) = L_C(M)$ holds.

Claim 1. $L_C(M) \subseteq L(B)$.

Proof. Let $w \in \Gamma^*$ be a word that belongs to the language $L_C(M)$. Thus, the computation of M that starts with the restarting configuration $(\lambda, \rhd w \lhd)$ is accepting. If $w = \lambda$, then $\delta_M(\rhd\lhd) = \mathsf{Accept}$, which implies that $q_F \in \Delta_B(q_0, \lambda)$. It follows that $w \in L(B)$ holds in this case.

Now assume that $w = w_1 w_2 \cdots w_n$ for some $n \geq 1$ and letters $w_1, \ldots, w_n \in \Gamma$. As $w \in L_C(M)$, we can now use the accepting computation of M for w to construct a representation as in the example above. This representation translates into a sequence of states of B, and it can be shown that this sequence of states yields an accepting computation of B for the input w. □

Claim 2. $L(B) \subseteq L_C(M)$.

Proof. We have to check that we can deduct a valid computation of M from an accepting computation of B. So let $w \in \Gamma^*$ be any word in $L(B)$, and let $n = |w|$. If $w = \lambda$, then $q_F \in \delta_B(q_0, \lambda)$, which implies that $\delta_M(\rhd\lhd) = \mathsf{Accept}$ holds, which in turn means that $w \in L_C(M)$.

If $w = w_1 \in \Gamma$, then there exist sequences $W_1 = (w_{1,1}, \ldots, w_{1,k_1})$ over Γ and $L_1 = (l_{1,1}, \ldots, l_{1,k_1-1})$ and $R_1 = (r_{1,1}, \ldots, r_{1,k_1-1})$ over $\mathbb{N}$ such that

- $w_{1,1} = w_1$,
- $((\Lambda, (\rhd), \Lambda), (L_1, W_1, R_1)) \in \Delta_B(q_0, w_1)$, and
- $q_F \in \Delta_B(((\Lambda, (\rhd), \Lambda), (L_1, W_1, R_1)), \lambda)$.

From the definition of Δ_B it follows that either $k_1 = 1$, and then $\mathsf{Accept} \in \delta_M(\rhd w_1 \lhd)$, or $k_1 > 1$, and then $l_{1,j} = 1 = r_{1,j}$ for all $j = 1, \ldots, k_1 - 1$, $w_{1,j+1} \in \delta_M(\rhd w_{1,j} \lhd)$ for all $j = 1, \ldots, k_1 - 1$, and $\mathsf{Accept} \in \delta_M(\rhd w_{1,k_1} \lhd)$. Hence, we see that the computation of M that begins with the restarting configuration $(\lambda, \rhd w \lhd)$ accepts, that is, $w = w_1 \in L_C(M)$.

Now assume that $w = w_1 \cdots w_n$ for some $n \geq 2$ and letters $w_1, \ldots, w_n \in \Gamma$. As B accepts on input w, there exist sequences $W_i = (w_{i,1}, \ldots, w_{i,k_i})$ over Γ and sequences of integers $L_i = (l_{i,1}, \ldots, l_{i,k_i-1})$ and $R_i = (r_{i,1}, \ldots, r_{i,k_i-1})$, $i = 1, \ldots, n$, such that all of the following conditions are met:

1. $((\Lambda, (\rhd), \Lambda), (L_1, W_1, R_1)) \in \Delta_B(q_0, w_1)$,
2. $((L_{i-1}, W_{i-1}, R_{i-1}), (L_i, W_i, R_i)) \in$
$\Delta_B((L_{i-2}, W_{i-2}, R_{i-2}), (L_{i-1}, W_{i-1}, R_{i-1})), w_i)$ for all $i = 2, \ldots, n$,
3. $q_F \in \Delta_B((L_{n-1}, W_{n-1}, R_{n-1}), (L_n, W_n, R_n)), \lambda)$.

From the definition of Δ_B we see that, for all $i = 1, \ldots, n$, $k_i \geq 1$ and $w_{i,1} = w_i$. Now let $N = N(R_1, \ldots, R_n) = \sum_{i=1}^{n} |R_i| = \sum_{i=1}^{n} (k_i - 1)$. By induction on N we will prove the following technical statement.

Claim 2.1. The computation of M that begins with the restarting configuration $(\lambda, \rhd w \lhd)$ consists of N cycles and an accepting tail, that is, it has the form

$$(\lambda, \rhd w \lhd) \vdash_M^c (\lambda, \rhd z^{(1)} \lhd) \vdash_M^c \cdots \vdash_M^c (\lambda, \rhd z^{(N)} \lhd) \vdash_M^* (\rhd u, v \lhd) \vdash_M \mathsf{Accept},$$

where $z^{(N)} = uv$ and $|v| = 2$.

Proof. If $N = 0$, then $k_i = 1$ for all $i = 1, \ldots, n$, and hence, $W_i = (w_i)$ and $L_i = R_i = \Lambda$ for all $i = 1, \ldots, n$. From the definition of Δ_B it follows that $\delta_M(w_{i-2} w_{i-1} w_i) = \mathsf{MVR}$ for all $i = 2, \ldots, n$, where $w_0 = \rhd$ is taken, and $\delta_M(w_{n-1} w_n \lhd) = \mathsf{Accept}$. Thus, the computation of M that begins with the restarting configuration $(\lambda, \rhd w \lhd)$ is simply an accepting tail computation.

Now assume that $N \geq 1$. Then $k_i > 1$ for some indices $i \in \{1, \ldots, n\}$, and accordingly, the corresponding sequences L_i and R_i are non-empty. Because of the consistency of the pairs (R_{i-1}, L_i), $i = 1, \ldots, n$, there exists an index $j \in \{1, \ldots, n\}$ such that $l_{j,1} = 1 = r_{j,1}$. Let $s \in \{1, \ldots, n\}$ be the minimal index such that $l_{s,1} = 1 = r_{s,1}$ holds. It follows that $k_s > 1$ and that $W_s = (w_{s,1}, w_{s,2}, \ldots, w_{s,k_s})$, where $w_s = w_{s,1} > w_{s,2}$. Let $\hat{w}$ denote the word $\hat{w} = w_1 \cdots w_{s-1} w_{s,2} w_{s+1} \cdots w_n \in \Gamma^n$. For this word the following result can be shown.

Claim 2.1.1. $(\lambda, \triangleright w \triangleleft) \vdash_M^c (\lambda, \triangleright \hat{w} \triangleleft)$.

We continue with the proof of Claim 2.1 by establishing the following claim, which will allow us to perform the intended inductive step.

Claim 2.1.2. The word $\hat{w}$ is accepted by the NFA B.

Proof. For all $i = 1, \ldots, n$, we define sequences $\hat{W}_i$ over Γ and sequences of integers $\hat{L}_i$ and $\hat{R}_i$ as follows:

$$\hat{W}_i = \begin{cases} (w_{i,2}, \ldots, w_{i,k_i}), & \text{if } i = s, \\ W_i, & \text{otherwise;} \end{cases}$$

$$\hat{L}_i = \begin{cases} (l_{i,2}, \ldots, l_{i,k_i-1}), & \text{if } i = s, \\ (l_{i,1} - 1, \ldots, l_{i,k_i-1} - 1), & \text{if } i = s+1, \\ L_i, & \text{otherwise;} \end{cases}$$

$$\hat{R}_i = \begin{cases} (r_{i,2}, \ldots, r_{i,k_i-1}), & \text{if } i = s, \\ (r_{i,1} - 1, \ldots, r_{i,k_i-1} - 1), & \text{if } i = s-1, \\ R_i, & \text{otherwise,} \end{cases}$$

and we take $\hat{k}_i$ to denote the length of the sequence $\hat{W}_i$, $i = 1, \ldots, n$. Then $\hat{k}_s = k_s - 1$, and $\hat{k}_i = k_i$ for all $i \neq s$. In order to unify the notation we write $\hat{w} = w_1 \cdots w_{s-1} w_{s,2} w_{s+1} \cdots w_n$ as $\hat{w} = \hat{w}_1 \cdots \hat{w}_n$. Also we write $\hat{W}_i$ as $\hat{W}_i = (\hat{w}_{i,1}, \ldots, \hat{w}_{i,\hat{k}_i})$, and $\hat{L}_i$ and $\hat{R}_i$ as $\hat{L}_i = (\hat{l}_{i,1}, \ldots, \hat{l}_{i,\hat{k}_i-1})$ and $\hat{R}_i = (\hat{r}_{i,1}, \ldots, \hat{r}_{i,\hat{k}_i-1})$, $i = 1, \ldots, n$. It can now be shown that the above sequences satisfy all of the following conditions::

1. $((\Lambda, (\triangleright), \Lambda), (\hat{L}_1, \hat{W}_1, \hat{R}_1)) \in \Delta_B(q_0, \hat{w}_1)$,
2. $((\hat{L}_{i-1}, \hat{W}_{i-1}, \hat{R}_{i-1}), (\hat{L}_i, \hat{W}_i, \hat{R}_i)) \in$
   $\Delta_B((\hat{L}_{i-2}, \hat{W}_{i-2}, \hat{R}_{i-2}), (\hat{L}_{i-1}, \hat{W}_{i-1}, \hat{R}_{i-1})), \hat{w}_i)$ for all $i = 2, \ldots, n$,
3. $q_F \in \Delta_B((\hat{L}_{n-1}, \hat{W}_{n-1}, \hat{R}_{n-1}), (\hat{L}_n, \hat{W}_n, \hat{R}_n)), \lambda)$.

It follows that the word $\hat{w}$ is accepted by B using the sequence of states defined above. As

$$N(\hat{R}_1, \ldots, \hat{R}_n) = \sum_{i=1}^{n} (\hat{k}_i - 1) = \sum_{i=1}^{n} (k_i - 1) - 1 = N(R_1, \ldots, R_n) - 1 = N - 1,$$

we can apply our induction hypothesis, which implies that the computation of M that begins with the restarting configuration $(\lambda, \rhd\hat{w}\lhd)$ consists of $N-1$ cycles and an accepting tail. Together with Claim 2.1.1 this says that the computation of M that begins with the restarting configuration $(\lambda, \rhd w\lhd)$ consists of N cycles and an accepting tail, which completes the proof of Claim 2.1. $\square$

From the claims above we obtain that $L(B) = L_C(M)$ holds. As M is deterministic, there is only a single accepting computation of B for each word $w \in L_C(M)$. It follows that B is unambiguous. $\square$

Claim 3. $|Q| \in 2^{O(|\Gamma|)}$.

Proof. The set Q of states of B contains the two designated states q_0 and q_F and certain states that consist of pairs of triples of the form (L, W, R), where W is a sequence of letters $W = (a_1, \ldots, a_m)$ from Γ such that $a_1 > \cdots > a_m$, and L and R are sequences of integers $L = (l_1, \ldots, l_{m-1})$ and $R = (r_1, \ldots, r_{m-1})$ such that $1 \leq l_1 \leq \cdots \leq l_{m-1}$ and $1 \leq r_1 \leq \cdots \leq r_{m-1}$. From upper bounds for the number of these sequences we will obtain an upper bound for the size of Q.

From the condition on the sequence W we see that $m \leq n = |\Gamma|$, and also $l_{m-1} \leq n$ and $r_{m-1} \leq n$. The sequence W defines the subset $\{w_1, \ldots, w_m\}$ of Γ, and different sequences W and W' yield different subsets. Hence, the number $2^n - 1$ of non-empty subsets of Γ is an upper bound for the number of different subsequences W.

The sequence L can be interpreted as a multiset over the set of integers $\{1, \ldots, n\}$, because it can contain repetitions. This multiset is of size at most $n-1$ (counting elements with their multiplicities). There are $\binom{n+r-1}{r}$ such multisets of size r (see, e.g., [14]), and hence, the number of possible sequences L is bounded from above by the expression

$$\sum_{r=0}^{n-1} \binom{n+r-1}{r} \leq \sum_{r=0}^{n-1} \binom{2n}{r} \leq \sum_{r=0}^{2n} \binom{2n}{r} = 2^{2n},$$

and the same is true for the number of possible sequences R. Hence, there are at most $2^{2n} \cdot 2^n \cdot 2^{2n} = 2^{5n}$ different triples of the form (L, W, R), and so the number of states of B is bounded from above by the number 2^{10n}. $\square$

It follows that B is of size $2^{O(n)}$. From B we now obtain an NFA A for the language $L(M) = L_C(M) \cap \Sigma^*$ by simply deleting all transitions from Δ_B that read a letter $x \in (\Gamma \smallsetminus \Sigma)$. Then it is immediate that A is an unambiguous NFA of size $2^{O(n)}$ that accepts the language $L(A) = L(B) \cap \Sigma^* = L(M)$. $\square$

For all $n \geq 3$, the language $U_n = \{a^{2^n}\}$ can be shown to be accepted by a stl-det-ORWW-automaton with an alphabet of $3n-1$ letters, while each NFA for U_n needs at least $2^n + 1$ states. Hence, the bound given in Theorem 5 is sharp up to the O-notation. In addition, we have the following consequence, which is a clear improvement over the upper bound given in Theorem 2 (b).

Corollary 6. *For each stl-det-ORWW-automaton M with alphabet of size n, there exists a DFA C of size $2^{2^{O(n)}}$ such that $L(C) = L(M)$ holds.*

4 Decision Problems for stl-det-ORWW-automata

The *emptiness problem* for an NFA $A = (Q, \Sigma, \delta, q_0, F)$ of size $|Q| = m$ is decidable nondeterministically in space $O(\log m)$ (see, e.g., [5]), and so, by Savitch's Theorem [13] it follows that $\mathsf{NFA\text{-}Emptiness} \in \mathsf{DSPACE}((\log |Q|)^2)$. Based on this observation we can use Theorem 5 to derive the following result.

Theorem 7. *The emptiness problem for stl-det-ORWW-automata is PSPACE-complete.*

Proof. Let $M = (\Sigma, \Gamma, \rhd, \lhd, \delta, >)$ be a stl-det-ORWW-automaton such that $|\Gamma| = n$. By Theorem 5, there exists an NFA A of size $2^{O(n)}$ such that $L(A) = L(M)$. Now we can check emptiness of $L(A)$ deterministically using space $(\log(2^{O(n)}))^2 = O(n^2)$. Thus, we see that $\mathsf{stl\text{-}det\text{-}ORWW\text{-}Emptiness} \in$ PSPACE.

Now let $A_1, \ldots, A_t$ be $t \geq 2$ DFAs over a common input alphabet Σ of size k such that A_i has n_i states, $1 \leq i \leq t$. From these DFAs we can construct a stl-det-ORWW-automaton M with a tape aphabet of size $k \cdot (1 + n_1 + \cdots + n_{t-1}) + n_t$ such that $L(M) = \bigcap_{i_1}^{t} L(A_i)$ [10]. Hence, M has at most $O((k \cdot \sum_{i=1}^{t} n_i)^3)$ transitions, and so it can be computed from $A_1, \ldots, A_t$ in polynomial time. Now $L(M) \neq \emptyset$ iff $L(A_1) \cap \cdots \cap L(A_t) \neq \emptyset$, which shows that the above construction yields a polynomial-time reduction from the $\mathsf{DFA\text{-}Intersection\text{-}Emptiness\ Problem}$ to $\mathsf{stl\text{-}det\text{-}ORWW\text{-}Emptiness}$. As the former is PSPACE-complete (see, e.g., [4]), we see that the latter is also PSPACE-hard. Together with the membership in PSPACE shown above, PSPACE-completeness follows. □

From this theorem we also get the following completeness results.

Corollary 8. *For stl-det-ORWW-automata, universality, finiteness, inclusion, and equivalence are PSPACE-complete.*

Proof. **Universality:** Let M be a stl-det-ORWW-automaton with input alphabet Σ. In polynomial time we can construct a stl-det-ORWW-automaton M^c for the language $L(M^c) = (L(M))^c = \Sigma^* \smallsetminus L(M)$ from M such that M^c uses the same tape alphabet as M [10]. The automaton M is universal, that is, $L(M) = \Sigma^*$, iff $L(M^c) = \emptyset$. PSPACE-completeness of the universality problem now follows from PSPACE-completeness for the emptiness problem.

Inclusion and Equivalence: Let M_1 and M_2 be stl-det-ORWW-automata with alphabets of sizes n_1 and n_2, respectively. In polynomial time we can construct a stl-det-ORWW-automaton M with an alphabet of size $O(n_1 \cdot n_2)$ from M_1 and M_2 such that $L(M) = L(M_1) \cap L(M_2)^c$ [10]. Now $L(M_1) \subseteq L(M_2)$ iff $L(M_1) \cap L(M_2)^c = \emptyset$ iff $L(M) = \emptyset$. It follows that the inclusion problem is in PSPACE, which in turn implies immediately that the equivalence problem is in PSPACE.

On the other hand, let M' be a stl-det-ORWW-automaton that accepts the empty set. Then $L(M) = L(M')$ iff $L(M) \subseteq L(M')$ iff $L(M) = \emptyset$. Thus, PSPACE-completeness of the inclusion and the equivalence problems follows from PSPACE-completeness for the emptiness problem.

Finiteness: Let $M = (\Sigma, \Gamma, \rhd, \lhd, \delta, >)$ be a stl-det-ORWW-automaton. We take a new symbol $\Box$, that is, $\Box \notin \Gamma$, and define a stl-det-ORWW-automaton $M' = (\Sigma', \Gamma', \rhd, \lhd, \delta', >)$ as follows:

- $\Sigma' = \Sigma \cup \{\Box\}$ and $\Gamma' = \Gamma \cup \{\Box\}$,
- the transition function δ' is obtained from δ by simply interpreting an occurrence of the symbol $\Box$ as an occurrence of the right delimiter $\lhd$.

Then $L(M') = L(M) \cup (L(M) \cdot \Box \cdot \Sigma'^*)$, which means that $L(M')$ is finite iff $L(M) = \emptyset$. PSPACE-hardness of finiteness now follows from PSPACE-hardness of the emptiness problem.

On the other hand, from a stl-det-ORWW-automaton M with an alphabet of size n we can construct an NFA A of size $2^{O(n)}$ such that $L(M) = L(A)$. Just like emptiness, also infiniteness is decidable for A nondeterministically in space $\log(2^{O(n)}) \in O(n)$, and hence, it is decidable deterministically in space $O(n^2)$. Thus, finiteness for stl-det-ORWW-automata is indeed PSPACE-complete. $\Box$

In the literature many subfamilies of the regular languages have been studied (see, e.g., [1,3,11]). Here we only consider some of them, beginning with the *strictly locally testable languages* of [8,15], but the corresponding problem can be stated for any subclass of REG.

A language $L \subseteq \Sigma^*$ is strictly k-testable for some $k \geq 1$ if $L \cap \Sigma^k \cdot \Sigma^* = (A \cdot \Sigma^* \cap \Sigma^* \cdot B) \smallsetminus \Sigma^+ \cdot (\Sigma^k \smallsetminus C) \cdot \Sigma^+$ for some finite sets $A, B, C \subseteq \Sigma^k$. For example, the language $(a+b)^*$ is strictly 1-testable, and the language $a(baa)^+$ is strictly 3-testable, but the language $(aa)^*$ is not strictly locally testable.

For each $k \geq 1$, if a language L is given through a DFA, then it is decidable in polynomial time whether or not L is strictly locally k-testable. Also it is decidable in polynomial time whether L is strictly locally testable [2]. We are interested in the corresponding variant of these problems in which the language considered is given through a stl-det-ORWW-automaton. Here we have the following result.

Theorem 9. *The following problem is PSPACE-complete for each $k \geq 1$:*

INSTANCE: A stl-det-ORWW-automaton M.
QUESTION: Is the language $L(M)$ strictly locally k-testable?

The construction in the proof shows that the problem of deciding strictly locally testability is at least PSPACE-hard for stl-det-ORWW-automata, but it remains open whether this problem is in PSPACE.

Using the same kind of reasoning it can be shown that, for a stl-det-ORWW-automaton, also the problems of deciding whether the accepted language is *nilpotent*, *combinatorial*, *circular*, *suffix-closed*, *prefix-closed*, *suffix-free*, or *prefix-free* (see, e.g., [1,3] for the definitions of these notions) are PSPACE-complete.

5 Concluding Remarks

We have shown that stl-det-ORWW-automata, although being deterministic devices, can provide exponentially more succinct representations for regular languages than NFAs. In addition, we have shown that many decision problems of interest are PSPACE-complete for stl-det-ORWW-automata. However, some open problems remain, for example:

– Can the given upper bounds be further improved by providing small constants in the exponents?
– Is the problem of deciding whether the language $L(M)$ that is accepted by a given stl-det-ORWW-automaton M is strictly locally testable decidable in polynomial space?

References

1. Bordihn, H., Holzer, M., Kutrib, M.: Determination of finite automata accepting subregular languages. Theor. Comp. Sci. **410**, 3209–3222 (2009)
2. Caron, P.: Families of locally testable languages. Theor. Comp. Sci. **242**, 361–376 (2000)
3. Dassow, J.: Subregular restrictions for some language generating devices. In: Freund, R., Holzer, M., Truthe, B., Ultes-Nitsche, U. (eds.) Proceedings of the Fourth Workshop on Non-Classical Models for Automata and Applications (NCMA 2012). books@ocg.at, Band, vol. 290, pp. 11–26. Oesterreichische Computer Gesellschaft, Wien (2012)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
5. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata - a survey. Inform. Comp. **209**, 456–470 (2011)
6. Hundeshagen, N., Otto, F.: Characterizing the regular languages by nonforgetting restarting automata. In: Mauri, G., Leporati, A. (eds.) DLT 2011. LNCS, vol. 6795, pp. 288–299. Springer, Heidelberg (2011)
7. Kutrib, M., Reimann, J.: Succinct description of regular languages by weak restarting automata. Inform. Comp. **206**, 1152–1160 (2008)
8. McNaughton, R.: Algebraic decision procedures for local testability. Math. Syst. Theor. **8**, 60–76 (1974)
9. Mráz, F., Otto, F.: Ordered restarting automata for picture languages. In: Geffert, V., Preneel, B., Rovan, B., Štuller, J., Tjoa, A.M. (eds.) SOFSEM 2014. LNCS, vol. 8327, pp. 431–442. Springer, Heidelberg (2014)
10. Otto, F.: On the descriptional complexity of deterministic ordered restarting automata. In: Jürgensen, H., Karhumäki, J., Okhotin, A. (eds.) DCFS 2014. LNCS, vol. 8614, pp. 318–329. Springer, Heidelberg (2014)
11. Pin, J.-E.: Syntactic semigroups. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, pp. 679–746. Springer, Berlin (1997)
12. Průša, D.: Weight-reducing Hennie machines and their descriptional complexity. In: Dediu, A.-H., Martín-Vide, C., Sierra-Rodríguez, J.-L., Truthe, B. (eds.) LATA 2014. LNCS, vol. 8370, pp. 553–564. Springer, Heidelberg (2014)
13. Savitch, J.E.: Relationships between nondeterministic and deterministic tape complexities. J. Comp. Syst. Sci. **4**, 177–192 (1970)
14. Stanley, R.P.: Enumerative Combinatorics, vol. 1, 2nd edn. Cambridge University Press, Cambridge (2012)
15. Zalcstein, Y.: Locally testable languages. J. Comp. Syst. Sci. **6**, 151–167 (1972)